Calhoun
Institutional Archive of the Naval Postgraduate School

Calhoun: The NPS Institutional Archive

DSpace Repository

Theses and Dissertations 1. Thesis and Dissertation Collection, all items

1968-12

A procedure for the synthesis of a fundamental loop or cut set matrix

Lohse, Hans Jurgen

Monterey, California. U.S. Naval Postgraduate School

http://hdl.handle.net/10945/12892

Copyright is reserved by the copyright owner.

Downloaded from NPS Archive: Calhoun

DUDLEY
KNOX
LIBRARY

http://www.nps.edu/library

Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

NPS ARCHIVE
1968
LOHSE, H.

A PROCEDURE FOR THE SYNTHESIS OF A FUNDAMENTAL LOOP OR CUT SET MATRIX

by

Hans Jürgen Lohse

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943-5101

UNITED STATES
NAVAL POSTGRADUATE SCHOOL

THESIS

A PROCEDURE FOR THE SYNTHESIS OF A FUNDAMENTAL LOOP OR CUT SET MATRIX

by

Hans Jürgen Lohse

December 1968

This document has been approved for public release and sale; its distribution is unlimited.

A PROCEDURE FOR THE SYNTHESIS OF A FUNDAMENTAL LOOP OR CUT SET MATRIX

by

Hans Jürgen Lohse
Lieutenant Commander, Federal German Navy
B.S., Naval Postgraduate School, 1967

Submitted in partial fulfillment of the
requirements for the degree of

ELECTRICAL ENGINEER

from the

NAVAL POSTGRADUATE SCHOOL
December 1968

NPS ARCHIVE
1968
LOHSE, H

~~Thesis L7914 c.1~~

ABSTRACT

Existing realization procedures for a fundamental loop or cut set matrix are reviewed, compared, and classified broadly on the basis of their underlying approach. A new combinatorial synthesis technique is presented utilizing the concepts of trunk branches, main branches, limbs, and unique connections which are introduced. This procedure is direct, easy to apply and learn, general, and yields an expression for the number of physically different or alternate realizations which are possible. A general computer program for realization of the graphs is presented and illustrated with some examples.

LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIF. 93940

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943-5101

TABLE OF CONTENTS

Section Page

INTRODUCTION 11

MATHEMATICAL REPRESENTATION OF ELECTRIC NETWORKS 14

2.1 Introduction 14

2.2 The Electric Network as a Graph 15

2.3 Description of a Graph in Matrix Form 15

2.3.1 The Incidence Matrix A 15

2.3.2 Trees and Co-trees 16

2.3.3 The Fundamental Loop Set Matrix B and the Fundamental Cut Set Matrix Q 18

2.4 General Network Analysis 21

2.4.1 Loop Analysis 22

2.4.2 Cut Set Analysis 23

2.4.3 State Variable Analysis 25

REVIEW OF EXISTING REALIZATION PROCEDURES FOR THE FUNDAMENTAL LOOP AND CUT SET MATRIX 30

3.1 The Realization Problem 30

3.2 The Linear Tree Port Structure Approach 31

3.3 Reducing the Cut Set Matrix to a Set of Incidence Matrices 35

3.4 Successive Removal of Tree Tips 39

3.5 Combining Loops within 2-Isomorphisms 44

3.6 Generating the Tree from Path Sets 49

3.7 Summary 50

REALIZATION OF THE LOOP MATRIX B USING THE COMBINATORIAL APPROACH 53

Section Page

4.1 Introduction 53

4.2 The Concept of Trunk, Main, and Limb Branches 53

4.3 Ordering Trunk Branches and the Concept of Unique Connections 55

4.4 An Illustrative Example 60

4.4.1 Partitioning the F Matrix 60

4.4.2 Establishing the Trunk 62

4.4.3 Making Unique Connections 64

4.4.4 Extending Main Branches 65

4.4.5 Connecting the Limbs 66

4.4.6 Connecting the Links 66

4.5 Further Examples 66

4.5.1 Graph without Limbs 66

4.5.2 Graph without Main Branches 68

4.5.3 Unrealizable Loop Matrix 68

4.5.4 General Graph with Separate Parts 70

PROGRAMMING THE REALIZATION PROCEDURE ON A DIGITAL COMPUTER 73

5.1 Introduction 73

5.2 Input/Output Considerations 73

5.3 Partitioning the F Matrix and Establishing the Trunk 74

5.4 Connecting Main Branches and Limbs 78

5.5 Connecting Links 79

5.6 Two Computer Solutions

Section Page

EVALUATION OF THE TOTAL NUMBER OF DIFFERENT GRAPHS HAVING AN IDENTICAL LOOP MATRIX 87

6.1 Introduction 87

6.2 The Concept of Different Graphs 87

6.3 Evaluation of the Total Number of Different Trunks 88

6.4 Evaluation of the Total Number of Different Main Branch Connections 91

6.5 Evaluation of the Total Number of Different Limb and Link Connections 92

6.6 Evaluation of the Total Number of Different Graphs 93

CONCLUSION AND SUGGESTION FOR FURTHER RESEARCH 95

BIBLIOGRAPHY 96

APPENDIX I: Glossary of Topological Terms 98

APPENDIX II: Transformation of Current and Voltage Sources 102

APPENDIX III: Computer Program 105

LIST OF ILLUSTRATIONS

Figure		Page
1.	Electric Network and Corresponding Graphs	19
2.	Separate Parts and Loops	19
3.	Cut Sets	19
4.	Trees	20
5.	Fundamental Loops and Cut Sets	20
6.	Typical Branch	28
7.	A Proper Tree	28
8.	Linear Tree Port Structure	36
9.	Graph used in Section 3.2	36
10.	Graph demonstrating Kim's Procedure	36
11.	Removal of a Cut Set	41
12.	Graph used in Section 3.3	41
13.	Graph demonstrating Mayeda's Procedure	41
14.	Illustration of Thinning	43
15.	Illustration of Shrinking	43
16.	Illustration of Reduction	43
17.	Realization Steps in Iri's Procedure	46
18.	Switching Network	46
19.	Isomorphic Graphs	48
20.	Non-Separated and Separable Graphs	48
21.	Combining Loops within 2-isomorphisms	48
22.	Realization Steps in Fu's and Löfgren's Procedure	51
23.	Demonstration of Gould's Procedure	51
24.	Illustration of Theorem 5	59

Figure Page

25. Trunk Branches, Main Branches, and Limbs 59

26. Illustration of Theorems 7a and 7b 59

27. Possible Trunks 67

28. Unique Connections 67

29. Graph of Section 4.4 67

30. Graph of Section 4.5.1 69

31. Graph of Section 4.5.2 69

32. Trunk of Section 4.5.3 69

33. Graph of Section 4.5.4 72

34. Graph of Section 5.2 81

35. Patterns of Link Positions 81

36. Graph of Section 5.6 86

37. Different Graphs 90

38. Different Trunks 90

39. Different Trunks 90

40. Different Connections of Main Branches 94

41. Different Connections of Limbs and Links 94

42. Voltage Source Transformation 104

43. Current Source Transformation 104

44. Typical Branch of an Electric Network 104

ACKNOWLEDGEMENT

I wish to thank my thesis advisor, Professor Dr. S. R. Parker, for his assistance. Not only did his lectures inspire me to investigate this subject but his experience and advice helped to shape many of the ideas presented in this thesis. I also like to thank my wife, Gesine, for her understanding that I had to compromise between studies, research, and family life.

INTRODUCTION

In 1847 Kirchhoff presented a systematic way to analyze complicated electric networks[19]. He related the topological structure, circuit elements, voltages and currents in the most general form. His results, except for the famous "Kirchhoff's Voltage Law" and "Kirchhoff's Current Law", were almost forgotten. There seemed to be no need for further investigations. However, during the last ten years engineers started to investigate properties of networks which previously appeared to be too theoretical and thus of no practical value. The reason for this being that many modern electric network problems cannot be solved by applying conventional analytical methods only. Among these problems are: The generation of equivalent networks with a minimum number of circuit components, optimum layout of electric networks for integrated circuits, and the design of switching networks and communication nets.

The first problem that was attacked by several investigators was the realization of a given fundamental loop or cut set matrix. These matrices play an important role in the formulation of Kirchhoff's general voltage and current laws and express the topological properties of the circuit. Since this theory treats the electric network as a graph many contributions were made by mathematicians. Each of the existing realization procedures in the literature suffers from some of the following limitations: It is either too abstract and cumbersome to be of practical value, it involves too much work and difficult matrix pattern recognitions on the part of the designer to be of value, or it fails in some cases.

In this thesis, Chapter 2 presents a summary of the mathematical representation of electric networks applying topological formulations as currently used for computer analysis. In Chapter 3 existing procedures for the realization of a fundamental loop or cut set matrix are reviewed, compared, and classified broadly according to their underlying ideas which in many cases were determined to be basically similar, a fact not always recognized in the literature. In Chapter 4 a new combinatorial synthesis is presented. This procedure introduces some new topological aspects and properties of a graph. Trunk branches, main branches, limbs and uniquely connected branches are introduced. The realization procedure presented is not only general and efficient, but also provides insight into the topological properties of an electric network. As a result the algorithms developed are directly applicable to computer programming. A detailed general realization program (the first available in the literature to the author's knowledge) is presented in Chapter 5. Moreover, this procedure is a sufficient condition for the realizability of a loop or cut set matrix. If this procedure yields a graph, i.e., if the loop or cut set matrix is realizable, the total number of physically different networks corresponding to a given loop or cut set matrix can be computed as presented in Chapter 6. This result has not been available previously.

MATHEMATICAL REPRESENTATION OF ELECTRIC NETWORKS

2.1 Introduction

The use of digital computers in the analysis and design of electric networks has not only provided a "super-sliderule" for numerical calculations, but has also directed the attention of engineers to such mathematical areas as linear algebra, topology, and state variable theory. Today classical methods such as Ohm's law, Kirchhoff's laws, and transform theory stated in terms of matrix theory and topology are the essential tools of applied electrical engineering. [1]

In this chapter it will be shown how to obtain a graph corresponding to an electric network. The most important notions of linear graph theory will then be defined and the concepts of fundamental loop and cut sets introduced. Next an algebraic description of a graph will be given in terms of the fundamental loop set matrix, B, or fundamental cut set matrix, Q.

Kirchhoff introduced many of these ideas in his original work on networks in 1847, at a time when little was known about topology[19]. During the last fifty years his results have been reexamined and generalized by men like Cauer, Foster, and Guillemin.

2.2 The Electric Network as a Graph

Given an electric network (Fig. 1a) made up of lumped circuit elements, one can find the associated linear graph* (Fig. 1b) by the following rules:

1. Open circuit each ideal current source.**
2. Short circuit each ideal voltage source.
3. Replace the passive elements by edges whose endpoints are called nodes.
4. For an oriented graph (Fig. 1c), redraw the graph maintaining the same configuration and include the directions of the currents.

2.3 Description of a Graph in Matrix Form

2.3.1 The Incident Matrix A

One way of describing the graph of Fig. 1 is to list all edges and nodes and to specify the two nodes between which each edge is connected. The element $a_{ij} = 1$ if edge i is incident to node j. When the graph is directed, $a_{ij} = \pm 1$ depending on the direction associated with edge i. By definition, $a_{ij} = +1$ if the ith element is directed away from the jth node. The undirected graph of Fig. 1 is listed in the matrix below.

---

* Since most terms are intuitively familiar, not all definitions are stated in the text, but are listed in Appendix AI.

** See Appendix AII for a detailed discussion of the topological representation of sources.

$$A = [a_{ij}] = \overset{\text{edges}}{\begin{array}{c|cccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline a & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ b & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ c & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ d & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ e & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{array}} \quad \text{nodes}$$

2.3.2 Trees and Co-trees

A graph can also be specified by its edges alone. Before doing so, some concepts of graph theory have to be introduced.

A graph is connected if there is at least one path along the edges between any two nodes. The graph of Fig.2c is connected, the graphs of Figs. 2a and 2b are not. The unconnected subgraphs are called separate parts. A single node also represents a separate part (Fig. 2b). An edge is removed if the line segment between the two nodes is deleted, the two nodes, however, remain part of the new graph. Removing edges 1 and 8 from Fig. 1b yields the graph in Fig. 2c. A subgraph of a graph is called a loop if

(a) the subgraph is connected,

(b) exactly two edges of the subgraph are incident with each node,

(c) exactly two nodes of the subgraph are connected to each edge.

Edges (1,5,6) of Fig. 2a form a loop

Edges (2,3,6,7,4,5) of Fig. 2c do not form a loop since there are 4 edges incident at node e.

A set of edges of a connected graph is called a cut set (Figs. 3b and 3c) if

(a) removal of all the edges of the cut set generates two separate parts.

(b) Removal of all but one element of the cut set leaves the remaining graph connected.

We can now introduce the concept of a tree. A tree of a connected graph is a connected subgraph which contains all the nodes of the graph, but does not have any loops.

The edges of the graph which are contained in the tree are called tree branches or just branches.

The edges of the graph which are not contained in the tree form the co-tree. They are called chords or links.

The same graph (Fig. 1b) can have different trees (Figs. 4a-c). The graph in Fig. 4d does not represent a tree.

Once we have specified a tree (T) for a connected graph of n nodes and e edges, then following properties hold:

(a) There is a unique path along the tree between any pair of nodes.

(b) There are $b = (n-1)$ tree branches and $l = e-(n-1)$ links, where n is the number of nodes and e the number of edges.

(c) Every link of T and the tree path between its nodes form a unique loop (fundamental loop).

(d) Every cut of the graph, which besides links cuts only one tree branch, forms a unique cut set of the graph (fundamental cut set).

2.3.3 The Fundamental Loop Set Matrix B and the Fundamental Cut Set Matrix Q

To find B and Q for the graph in Fig. 1b we first choose a tree, say that of Fig. 4b. The columns of the fundamental loop set matrix, B, correspond to the edges of the graph, and the rows to the fundamental loops (which are determined by the links).

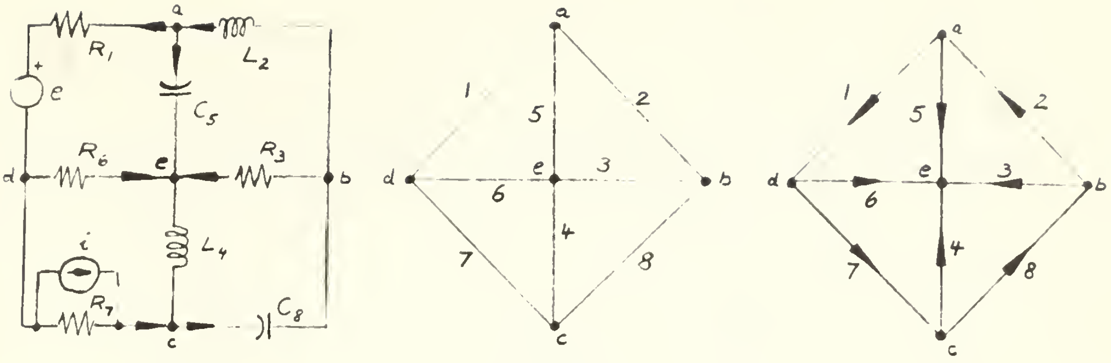
The reference directions of each fundamental loop shall be that of the defining link as indicated in Fig.5 . If edge k is not in loop i, $b_{ik} = 0$ If the direction of edge k is opposite to the reference direction of loop i, then $b_{ik} = -1$. For Fig. 5a

$$
B = \begin{array}{cccccccc|l}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \\
\hline
1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & -1 & 1 & 1 & 2 \quad \ell \text{ loops} \\
0 & 0 & 1 & 0 & 0 & -1 & 1 & 1 & 3 \\
0 & 0 & 0 & 1 & 0 & -1 & 1 & 0 & 4 \\
\multicolumn{4}{c}{\text{links}} & \multicolumn{4}{c}{\text{branches}} &
\end{array} \qquad (1.1)
$$

B can always be partitioned as

$$B = \left[U_{\ell} \quad F \right] \qquad (1.2)$$

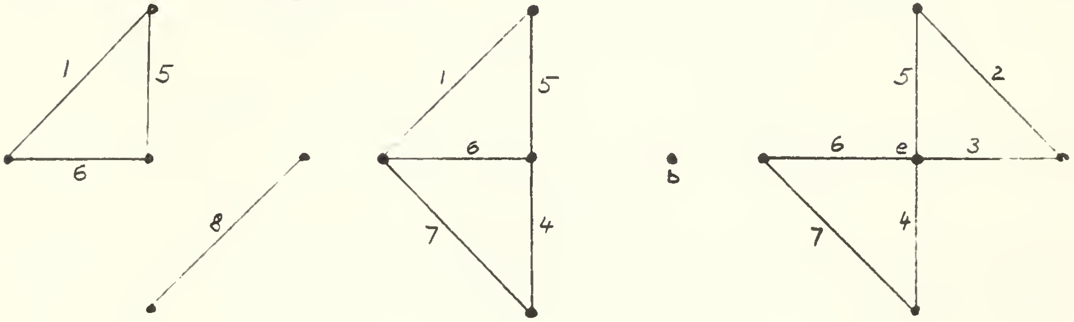
Where U_{ℓ} is the $\ell \times \ell$ identity matrix and F is a $b \times \ell$ matrix. The columns of the fundamental cut set matrix, Q correspond to the edges of the graph (as was the case in the B matrix); the rows, however, correspond to the fundamental cut sets (determined by the tree branches). The



(a) (b) (c)

Figure 1

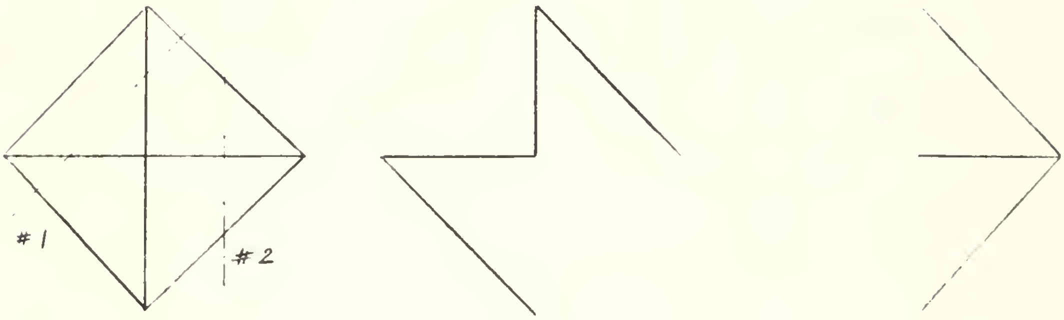
An electric network (a) and the corresponding undirected (b) and directed graphs (c).



(a) (b) (c)

Figure 2

Separate parts (a,b) of a graph, subgraph (c) which is not a loop.



(a) (b) (c)

Figure 3

Cut sets, #1 in (b) and #2 in (c), of a graph (a).

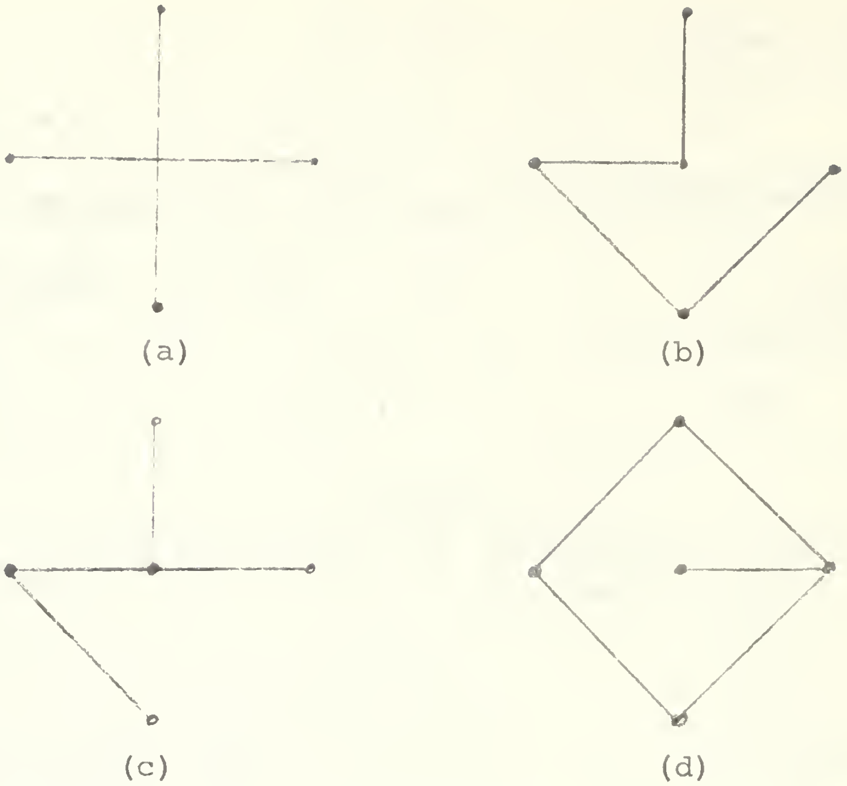


Figure 4

Trees (a,b,c) of graph in Fig.1, subgraph (d) which is not a tree.

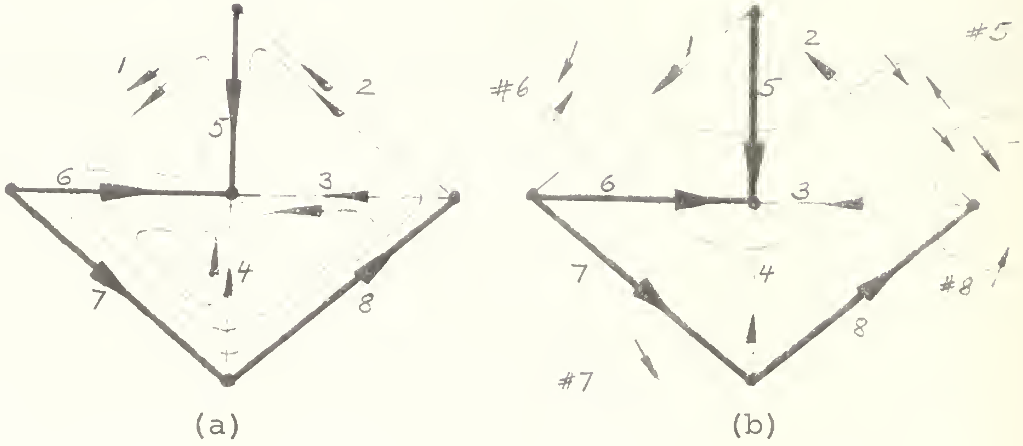


Figure 5

Fundamental loops (a) and cut sets (b) of a graph.

reference direction of each fundamental cut set shall be that of the defining tree branch as indicated by the short arrows in Fig.5b. If edge k is not in the cut set i, $q_{ik} = 0$, if the direction of edge k is opposite to the reference direction of cut set i, $q_{ik} = -1$. For Fig. 5b

$$Q = \begin{array}{cccc|cccc|c} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \\ \hline 1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 5 \\ -1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 6 \\ 0 & -1 & -1 & -1 & 0 & 0 & 0 & 1 & 7 \\ 0 & -1 & -1 & 0 & 0 & 0 & 0 & 1 & 8 \\ \hline \multicolumn{4}{c|}{\text{links}} & \multicolumn{4}{c|}{\text{branches}} & \end{array} \quad \text{b cut set} \tag{1.3}$$

Q can always be partitioned as

$$Q = \left[E \mid U_b \right] \tag{1.4}$$

Following relation holds (see Section 2.4.2. for the proof).

$$E = -F^T$$

2.4 General Network Analysis

In classical network analysis two methods serve to analyze an electric network. For loop analysis the loop currents are chosen as variables and Kirchhoff's Voltage Law (KVL) is applied to each loop. For nodal analysis the node-to-datum voltages are chosen as variables and Kirchhoff's Current Law (KCL) is applied to the n nodes.

These classical methods shall be generalized and a third method, the state variable method, be introduced. To simplify the presentation we will assume that the impedance of each edge is resistive.

2.4.1 Loop Analysis

The response of a network is completely known if the voltages across and the currents in all its e edges are known. If, however, the impedance of the edges are known, then either the set of e edge voltages or e edge currents suffices, since these two sets are related by Ohm's Law.

Once a tree is chosen, fewer than e variables suffice to characterize the network. It has been shown that the removal or opening of all links does not leave any loops. Expressed in circuit terminology, setting the link currents equal to zero forces all the tree branch currents to be zero (KCL). Hence the set of ℓ link current (j) can be represented as a linear combination of the link currents(i).

It can be seen by inspection that

$$\underline{j} = B^T\underline{i} \tag{1.5}$$

where the components of the vectors $\underline{j}$ and $\underline{i}$ correspond to edge and link currents respectively.

Applying KVL to each loop the following equation is obtained

$$B\underline{v} = \underline{0} \tag{1.6}$$

where the components of $\underline{v}$ correspond to edge voltages. Representing a typical edge as in Fig. 6 and applying Ohm's Law equation (1.7) is obtained.

$$\underline{v} = R\underline{j} + \underline{v}_s - R\underline{j}_s \tag{1.7}$$

where R is an (e x e) diagonal matrix, r_{ii} representing the resistance of the ith edge, and $r_{ij} = 0$ for $i \neq j$. $\underline{j}$, $\underline{j}_s$, $\underline{v}$, and $\underline{v}_s$ are (e x 1) column vectors, whose ith component represents current, current source, voltage, and voltage source, respectively, of the ith edge.

Combining equations (1.5), (1.6), and (1.7) yields

$$BRB^t\underline{i} = - B\underline{v}_s + BR\underline{j}_s$$

or

$$Z_\ell \underline{i} = \underline{e}_s \tag{1.8}$$

where

$$Z_\ell \triangleq BRB^t \tag{1.9}$$

is called the loop impedance matrix, and is of order ℓ and

$$\underline{e}_s \hat{=} - B\underline{v}_s + BR\underline{j}_s \tag{1.10}$$

is called the loop voltage vector (ℓ x 1).

2.4.2 Cut Set Analysis

It has been shown that a tree connects all the nodes. If the tree branch voltages are forced to zero (short circuiting the tree branches), then all nodes coalesce, forcing all edge voltages to zero. Thus the set of $b = (n-1)$ tree branch voltages is the only independent set of edge voltages, i.e. each edge voltage (v) can be represented as a linear combination of the tree branch voltages(e).

It can be seen by inspection that

$$\underline{v} = Q^t\underline{e} \tag{1.11}$$

where the components of the vectors $\underline{v}$ and $\underline{e}$ correspond to edge and tree branch voltages respectively.

Applying KCL to each cut set,

$$Q\underline{j} = \underline{0} \ . \qquad (1.12)$$

Using Ohm's Law for each edge,

$$j = G\underline{v} + j_s - G\underline{v}_s \qquad (1.13)$$

where G is an (e x e) diagonal matrix, g_{ii} representing the conductance of the ith edge, $g_{ij} = 0$ for $i \neq j$. $\underline{j}, \underline{j}_s$ and $\underline{v}, \underline{v}_s$ represent the currents, voltages and sources of each edge as in equation (1.6).

Combining equations (1.11), (1.12), (1.13) yields

$$QGQ^t\underline{e} = QG\underline{v}_s - Q\underline{j}_s$$

or

$$Y_q\underline{e} = \underline{i}_s \qquad (1.15)$$

where

$$Y_q = QGQ^t \qquad (1.16)$$

is called the cut set admittance matrix and

$$\underline{i}_s = QG\underline{v}_s - Q\underline{j}_s \qquad (1.17)$$

is called the cut set current source vector.

Properties of the B and Q matrices which are important for the realization procedures described later in this thesis follow.

Theorem 1:

$$QB^t = 0 \qquad (1.18)$$

Proof: From equations (1.5) and (1.10) we have

$$B\underline{v} = \underline{0} \quad \text{and} \quad \underline{v} = Q^t\underline{e} \ .$$

Thus

$$BQ^t\underline{e} = \underline{0} \ .$$

Setting any $e_i = 1$ and all the other components of $\underline{e}$ equal to zero requires that each element of BQ^t equals zero.

Using equations (1.2) and (1.4) the following relation holds:

Theorem 2:

$$E = - F^t$$

Proof:

$$QB^t = \left[E \mid U_b \right] \left[\begin{array}{c} U_\ell \\ \hline F^t \end{array} \right] = \underline{0} \qquad (1.19)$$

$$E + F^t = \underline{0}$$

Thus the fundamental loop and cut set matrices can be obtained from one another by the relation

$$B = \left[U_\ell \mid F \right] \quad \text{and} \quad Q = \left[-F^t \mid U_b \right] \qquad (1.20)$$

2.4.3 State Variable Analysis

The description of electric networks by state variables is included here because its formulation rests heavily on the concept of trees, co-trees, and of network graphs. The advantages of state-variable analysis are [4]:

1. It represents a unified approach to networks having non-linear and time-varying elements.
2. It provides insight in the behavior of networks in such areas as sensitivity and stability.

3. It is better suited for digital computation since an nth order differential equation is represented by a set of n first order differential equations.

Definition: [5] The state of a system is the minimum set of variables, the state variables, which contain sufficient information about the past history of the system to permit the computation of all future states, given that the future inputs and the state equations of the system are known.

This definition does not attach any physical significance to a state variable, and it is up to the engineer to make a proper choice. One choice of state variables for an RLC network consists of capacitance voltages and inductance currents (assuming there are no loops consisting of capacitances only and no cut sets, which are made up completely of inductances) [6]:

This choice seems quite legitimate, since independent inductors and capacitors are the only energy storing elements. An example shall illustrate how the state equations are found. A linear, time-invariant network (Fig. 7a) is chosen, which does not contain C-loops, nor L-cut sets. The method can be modified to be applicable to these cases [7].

Choose a tree (Fig. 7b) that includes all capacitive edges and as many resistive edges as necessary. This tree is called a proper tree [6]. The links then will be made up of inductive edges and the remaining resistive edges. Using KVL, $B\underline{v} = \underline{0}$, it is convenient to partition $\underline{v} = [\underline{v}_R \mid \underline{v}_L \mid \underline{v}_C \mid \underline{v}_G]^T$, where the subvectors $\underline{v}_R$, $\underline{v}_L$, $\underline{v}_C$, and $\underline{v}_G$ correspond to the voltages of resistive and inductive links and capacitive and resistive tree branches, respectively. Thus

$$B\underline{v} = \underline{0} \qquad \text{or} \qquad \left[U_\ell \,\vdots\, F \right] \begin{bmatrix} \underline{v}_R \\ \underline{v}_L \\ \underline{v}_C \\ \underline{v}_G \end{bmatrix} = \underline{0}$$

For the example given:

$$B = \begin{array}{cccc|cccc c} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 2 \\ 0 & 0 & 1 & 0 & 0 & -1 & -1 & -1 & 3 \\ 0 & 0 & 0 & 1 & 0 & -1 & -1 & 0 & 4 \end{array}$$

$$\underline{v}_R = \begin{bmatrix} v_{R1} - v_{S1} \\ v_{R2} \end{bmatrix} \quad \underline{v}_L = \begin{bmatrix} v_{L3} \\ v_{L4} \end{bmatrix} \quad \underline{v}_C = \begin{bmatrix} v_{C5} \\ v_{C6} \end{bmatrix} \quad \underline{v}_G = \begin{bmatrix} v_{R7} \\ v_{R8} \end{bmatrix}$$

Using the same partitioning we write KCL as

$$Q\underline{j} = 0 \text{ or } \left[-F^t \,\vdots\, U_b \right] \begin{bmatrix} \underline{j}_R \\ \underline{j}_L \\ \underline{j}_C \\ \underline{j}_G \end{bmatrix} = \underline{0}$$

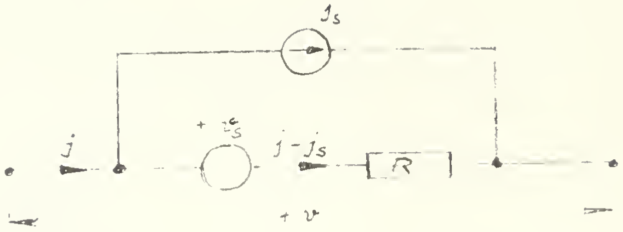
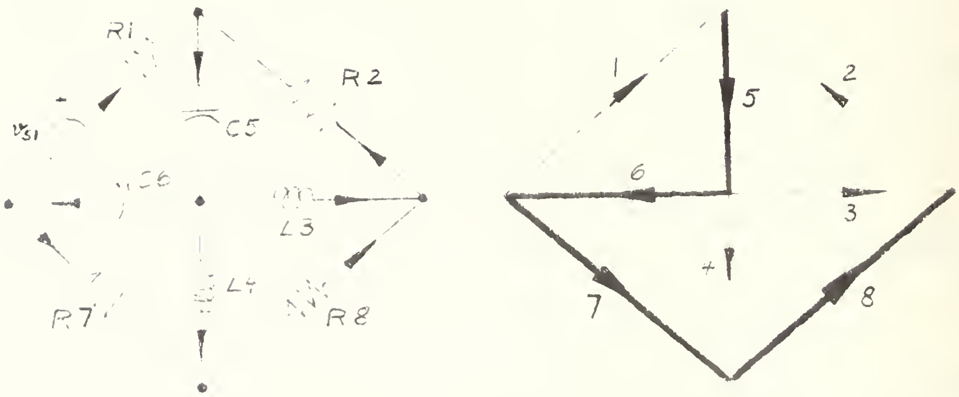


Figure 6

Typical branch of an electric network



(a) (b)

Figure 7

Proper tree for state variable representation

For the example given:

$\underline{j}_R = \begin{bmatrix} j_{R1} \\ j_{R2} \end{bmatrix}$ and $\underline{j}_L, \underline{j}_C, \underline{j}_G$ having the same form as $\underline{v}_L, \underline{v}_C, \underline{v}_G$ substituting j for v.

To eliminate the unwanted variables ($\underline{v}_R, \underline{v}_L, \underline{v}_G, \underline{j}_R, \underline{j}_C, \underline{j}_G$) we apply Ohm's Law:

$$\underline{v}_R = R\underline{j}_R + \underline{e}_R \qquad \underline{j}_G = G\underline{v}_G + \underline{i}_G$$

$$\underline{v}_L = L\frac{d}{dt}\underline{j}_L + \underline{e}_L \qquad \underline{j}_C = C\frac{d}{dt}\underline{v}_C + \underline{i}_C$$

Where $\underline{e}$ and $\underline{i}$ stand for independent sources.

$$R = \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix} \quad L = \begin{bmatrix} L_3 & 0 \\ 0 & L_4 \end{bmatrix} \quad G = \begin{bmatrix} G_7 & 0 \\ 0 & G_8 \end{bmatrix} \quad C = \begin{bmatrix} C_5 & 0 \\ 0 & C_6 \end{bmatrix}$$

Combining KVL, KCL and Ohm's Law, we obtain the state variable equations in normal form

$$\frac{d}{dt}\begin{bmatrix} C & 0 \\ 0 & L \end{bmatrix}\begin{bmatrix} \underline{v}_C \\ \underline{j}_L \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}\begin{bmatrix} \underline{v}_C \\ \underline{j}_L \end{bmatrix} + B\begin{bmatrix} \underline{i}_C \\ \underline{i}_G \\ \underline{e}_R \\ \underline{e}_L \end{bmatrix} \tag{1.21}$$

where A_{ij} and B are expressible in closed form in terms of the matrices F, R, L, G, and C [4].

The solution of the normal form (1.21) can be found in a number of ways using a digital computer [8]. Any set (y) of currents or voltages of the network can then be expressed in terms of the state variables ($\underline{x}$) and the sources ($\underline{u}$) as:

$$\underline{y} = D\underline{x} + E\underline{u} \tag{1.22}$$

REVIEW OF EXISTING REALIZATION PROCEDURES FOR THE FUNDAMENTAL LOOP AND CUT SET MATRIX

3.1 The Realization Problem

It has been shown that the fundamental loop set matrix B and cut set matrix Q are essential in the general analysis of electric networks. Moreover, switching circuits [15] and communication nets [3] can also be described conveniently by these matrices. A next logical step is to use the loop or cut set matrix in a synthesis procedure [15] and [3].

It was relatively easy to find the B and Q matrix for a given network. To realize a graph corresponding to B or Q, however, is not a trivial problem because there may exist no graph at all, there may exist exactly one graph, or there may exist many graphs having the same B or Q matrix. An exhaustive search procedure forbids itself. Since 1958 [16] several systematic synthesis procedures have been developed, which can be classified broadly according to their underlying ideas. In this chapter the same synthesis problem is solved using a method typical of each classification, so as to compare the procedures and the work required. The important theorems used in these procedures are illustrated. Their exact proofs can be found in the references as noted.

3.2 The Linear Tree Port Structure Approach

Using the properties of the admittance matrix Y of order n of a linear tree port structure*, Biorci [9] and Kim [10] find the corresponding graph having (n + 1) nodes and n ports. Since $Y = QGQ^t$ (1.16), this procedure can be used to realize a fundamental cut set matrix by setting all edge conductances equal to one. In Fig. 8, using edges 1, 2, and 3 as tree branches for ports 1, 2, and 3, the corresponding fundamental cut set matrix is

$$Q = \begin{array}{c|ccc|ccc} & 11 & 12 & 13 & 22 & 23 & 33 \\ \hline 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 2 & 0 & 1 & 1 & 1 & 1 & 0 \\ 3 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}$$

Forming $Y = QGQ^t$ the elements of row 1 of Y are $y_{11} = g_{13} + g_{12} + g_{11}$, $y_{12} = g_{13} + g_{12}$, $y_{13} = g_{13}$.

Assuming positive edge conductance, we observe, that the value of the elements of Y taper off to the right. Thus

$$y_{11} \geq y_{12} \geq \dots \geq y_{1n}$$

$$y_{22} \geq \dots \geq y_{2n}$$

....

Similarly

$$y_{nn} \geq y_{n-1,n} \geq \dots y_{1n}$$
$$y_{n-1,n} \geq y_{n-2,n} \geq \dots y_{n-1,n}$$

---

* Tree branches from a linear path and the terminals of each port are identified by the end nodes of a tree branch.

These observations can be generalized [3] into the following:

Theorem 3: The admittance matrix Y of a resistive n-port (with a linear tree port structure) in which the polarity of the port voltage and ordering of each port is aligned in one direction (Fig. 8), is a uniformly tapered matrix.

A matrix is uniformly tapered if

1. Each element is greater than or equal to zero

$$y_{ij} \geq 0, \text{ for all } i, j,$$

2. The magnitudes of the elements of each row of Y taper off

   (a) from the main diagonal to the right hand side

   $$y_{ij} \geq y_{i,j+1}, \text{ for } j \geq i$$

   (b) from the main diagonal to the top, the diagonal element being the largest one.

   $$y_{ij} \geq y_{i-1,j}, \text{ for } 1 < i \leq j$$

3. $y_{ij} + y_{i-1,j+1} \geq y_{i-1,j} + y_{i,j+1}$ for all $i \geq j$

Kim's method works best for a linear tree, Biorci's method works more efficient if the tree does not form a linear path. Kim's method is used to solve an example. His realization procedure consists of four steps:

1. Form $Y = QQ^t$
2. Take the column of Q with the largest number of ones. Assume the ones are located in rows

$i, j, \ldots, k$. Form the submatrix $Y_{i,j,\ldots,k}$ by retaining the ith, jth, ... kth rows and columns of Y and deleting the others. Place $Y_{i,j,\ldots k}$ in a uniformly tapered form. This is done by noting the following

(a) Interchanging row and column j with row and column m in $Y_{i,j,\ldots k}$ corresponds to interchanging the labeling of tree branches j and m.

(b) Multiplying all elements of row j and column j by (-1) means reversing the polarity of the jth tree branch, i.e. the jth port. (Note the element y_{jj} will remain positive).

3. Repeat step 2 for the columns of Q with successively smaller number of ones.

4. Group the tree branches together.

Example: Realize the given cut set matrix Q using Kim's method.

$$Q = \begin{array}{c|cccc:cccc} & 2 & 5 & 7 & 8 & 1 & 3 & 4 & 6 \\ \hline 1 & -1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 4 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 6 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}$$

$$QQ^t = Y = \begin{array}{c|cccc} & 1 & 3 & 4 & 6 \\ \hline 1 & 3 & -1 & 0 & -2 \\ 3 & -1 & 3 & -1 & 1 \\ 4 & 0 & -1 & 3 & 1 \\ 6 & -2 & 1 & 1 & 4 \end{array}$$

Column 2 of Q has the maximum number of non-zero elements.

$$Y_{1,3,6} = \begin{array}{c|ccc} & 1 & 3 & 6 \\ \hline 1 & 3 & -1 & -2 \\ 3 & -1 & 3 & 1 \\ 6 & -2 & 1 & 4 \end{array}$$

This matrix is not uniformly tapered since it violates condition 1, i.e. there are negative elements in $Y_{1,3,6}$.

Multiplying row and column 1 by (-1) yields

$$Y_{1,3,6} = \begin{array}{c|ccc} & 1 & 3 & 6 \\ \hline 1 & 3 & 1 & 2 \\ 3 & 1 & 3 & 1 \\ 6 & 2 & 1 & 4 \end{array}$$

This matrix is still not uniformly tapered since it violates conditions 2a and 2b.

Interchanging rows 3 and 6 and columns 3 and 6 (rule a)

$$Y_{1,6,3} = \begin{array}{c|ccc} & 1 & 6 & 3 \\ \hline 1 & 3 & 2 & 1 \\ 6 & 2 & 4 & 1 \\ 3 & 1 & 1 & 3 \end{array}$$

which is uniformly tapered.

Proceeding with column 5 of Q, yields:

$$Y_{1,6} = \begin{array}{c|cc} & 1 & 6 \\ \hline 1 & 3 & -2 \\ 6 & -2 & 4 \end{array}$$

placing in uniformly tapered form

$$Y_{1,6} = \begin{array}{c|cc} & 1 & 6 \\ \hline 1 & 3 & 2 \\ 6 & 2 & 4 \end{array}$$

Columns 7 and 8 yield

$$Y_{6,4} = \begin{array}{c|cc} & 6 & 4 \\ \hline 6 & 3 & 1 \\ 4 & 1 & 4 \end{array}$$

and

$$Y_{3,4} = \begin{array}{c|cc} & 3 & 4 \\ \hline 3 & 3 & -1 \\ 4 & -1 & 3 \end{array}$$

placing in uniformly tapered form

$$\begin{array}{c|cc} & 3 & 4 \\ \hline 3 & 3 & 1 \\ 4 & 1 & 3 \end{array}$$

The linear sub-trees (Fig.10) are then combined to form the tree as in Fig. 9.

It can be seen that this method is very difficult because the process of obtaining the uniformly tapered form is essentially a trial and error procedure.

3.3 Reducing the Cut Set Matrix to a Set of Incidence Matrices

Tutte [11] and Mayeda [12] derive a set of incidence matrices from a given cut set matrix and then combine their corresponding subgraphs. Auslander's [13] method arrives at similar results using the loop concept to justify the incidence relations. Tutte bases his algorithm on the theory of matroids*. He approaches the realization problem in much the same way as Mayeda. The basic ideas underlying their methods are as follows:

Connect all edges of a fundamental cut set, i (Fig. 11a), at an auxiliary node, n (Fig.11b). Then split this node, thus generating two separate graphs G_1 and G_2. Removal of the cut set, i, from G_1 and G_2 yields two subgraphs G_a and G_b. (Fig. 11c). The removal of the cut set, i, from the graph corresponds to omitting row i from Q and all columns

---

* A binary matroid is the class of elementary chains of a binary chain-group, i.e. the edges of a fundamental cut set form an elementary chain, all fundamental cut sets with respect to a given tree form a binary chain-group, thus constituting a class or binary matroid. This binary matroid can be represented by a matrix, which is called Q in this thesis.

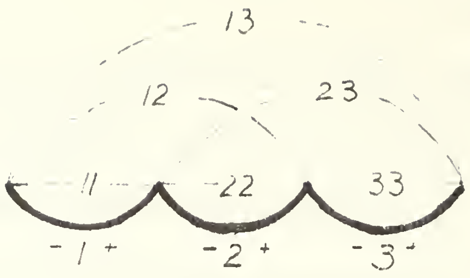


Figure 8

Linear tree port structure of a graph.

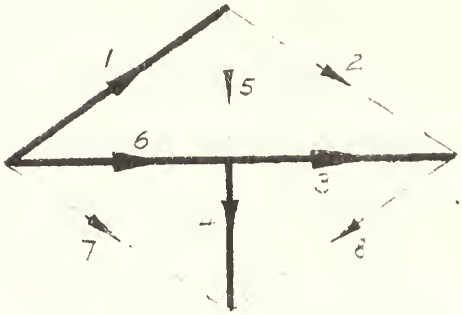


Figure 9

Graph of example in Section 3.2.

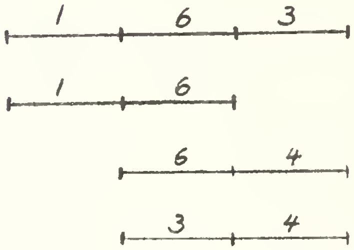


Figure 10

Subtrees found by Kim's procedure.

of Q that have a one in this row. This operation yields a new matrix H. If H can be partitioned as

$$H = \begin{bmatrix} H_1 & 0 \\ 0 & H_2 \end{bmatrix}$$

the cut set i does not represent an incidence set, that is the elements of i do not converge on a single node. Thus both separate parts G_a and G_b contain at least two nodes, one tree branch, and one link. H_1 and H_2 are the fundamental cut set matrices of G_a and G_b, respectively.
The fundamental cut set matrix $M_1(i)$ of G_1 is found by removing from Q all columns and rows which belong to H_2. Similarly $M_2(i)$ of G_2 is found. If H cannot be partitioned, one of the separate parts has to consist of only one node, indicating that the cut set i is an incidence set. In this case $M_1(i) = Q$ and $M_2(i) =$ row i, omitting the zero entries. Proceed by successively removing the other rows until all M-matrices are incidence matrices. If Q has b rows, (b+1) of these "minimum M-submatrices" [12] are obtained, yielding (b+1) subgraphs. These subgraphs are then combined by joining them at the auxiliary nodes.
Example: Realize the following Q matrix using Mayeda's procedure. The corresponding graph is shown in Fig. 12.

$$Q = \begin{array}{c|cccc|cccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline 5 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 6 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 7 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 8 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}$$

Underlining indicates minimum matrix.

Remove cut set $i = 5$ from Q

$$H = \begin{array}{c|ccccc} & 3 & 4 & 6 & 7 & 8 \\ \hline 6 & 1 & 1 & 1 & 0 & 1 \\ 7 & 1 & 1 & 0 & 1 & 0 \\ 8 & 1 & 0 & 0 & 0 & 1 \end{array}$$

Partitioning is not possible.

Letting $M_1(5) = Q$ and $\underline{M_2(5)} = \begin{array}{c|ccc} & 1 & 2 & 5 \\ \hline 5 & 1 & 1 & 1 \end{array}$, $M_2(5)$ represents a minimum M-matrix.

Proceeding with $M_1(5)$ cut set $j = 6$ is removed.

$$H = \begin{array}{c|ccc} & 5 & 7 & 8 \\ \hline 5 & 1 & 0 & 0 \\ 7 & 0 & 1 & 0 \\ 8 & 0 & 0 & 1 \end{array}$$

Omitting rows and columns 7 and 8 in Q yields:

$$\underline{M_1(56)} = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 5 & 1 & 1 & 0 & 0 & 1 & 0 \\ 6 & 1 & 1 & 1 & 1 & 0 & 1 \end{array}$$

Omitting row and column 5 in Q yields:

$$M_1(6) = \begin{array}{c|ccccccc} & 1 & 2 & 3 & 4 & 6 & 7 & 8 \\ \hline 6 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 7 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \end{array}$$

Proceeding with $M_1(6)$ cut set $k = 7$ is removed.

k = 7

$$\underline{M_{11}(67)} = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 6 & 7 \\ \hline 6 & 1 & 1 & 1 & 1 & 1 & 1 \\ 7 & 0 & 1 & 1 & 1 & 0 & 1 \end{array}$$

$$\underline{M_{1}(7)} = \begin{array}{c|ccccc} & 2 & 3 & 4 & 7 & 8 \\ \hline 7 & 1 & 1 & 1 & 1 & 0 \\ 8 & 1 & 1 & 0 & 0 & 1 \end{array}$$

Finally cut set l = 8 is removed from $M_1(6)$.

$\ell = 8$ $\quad M_{11}(68) = M_1(6)$

$$\underline{M_{1}(8)} = \begin{array}{c|ccc} & 2 & 3 & 8 \\ \hline 8 & 1 & 1 & 1 \end{array}$$

The realization steps are shown graphically in Fig. 13.

3.4 Successive Removal of Tree Tips

The methods of Guillemin [14] and Iri [15] successively remove the outermost branches of a tree ("tips"[14]), once these have been identified. Since Iri's method is more general (Guillemin's procedure fails in the case of star trees) its principles shall be outlined. The main idea is that only those edges are removed which do not make the resulting graph separable. This is achieved by three operations.

Thinning (Fig. 14) which means to delete a column with only one non-zero element or to delete all but one of a set of identical columns. This removes parallel links.

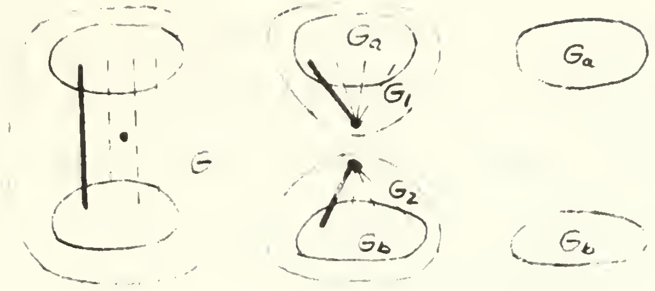
$$-F^t = \begin{array}{c|ccc} & 3 & 4 & 5 \\ \hline 1 & 1 & 1 & 0 \\ 2 & 1 & 1 & 1 \end{array} \quad \text{becomes} \quad \begin{array}{c|c} & 3 \\ \hline 1 & 1 \\ 2 & 1 \end{array}$$

Shrinking (Fig. 15), which means to delete rows (1 and 2), which have only one non-zero element in the same column. If there are identical rows (4,6,7), one interchanges the role of a link (10) and a tree branch (4), corresponding to one of these rows. The result is that all but one of these rows have only one non-zero element in a common column (column 4). The branches corresponding to these rows are "perfectly series" [15], i.e. they are in series in any 2-isomorphic graph (cf. sect. 3.5)

$$-F^t = \begin{array}{r|ccc} & 8 & 9 & 10 \\ \hline 1 & 1 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 0 & 0 & 1 \\ 4 & 0 & 1 & 1 \\ 5 & 1 & 1 & 1 \\ 6 & 0 & 1 & 1 \\ 7 & 0 & 1 & 1 \end{array} \qquad \begin{array}{r|ccc} & 8 & 9 & 4 \\ \hline 1 & 1 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 0 & 1 & 1 \\ 10 & 0 & 1 & 1 \\ 5 & 1 & 1 & 1 \\ 6 & 0 & 0 & 1 \\ 7 & 0 & 0 & 1 \end{array} \quad \text{becomes} \quad \begin{array}{r|ccc} & 8 & 9 & 4 \\ \hline 2 & 1 & 0 & 0 \\ 3 & 0 & 1 & 1 \\ 10 & 0 & 1 & 1 \\ 5 & 1 & 1 & 1 \\ 6 & 0 & 0 & 1 \end{array}$$

The left hand form of $-F^t$ corresponds to Fig. 15a, the right hand form to Fig. 15c, the one in the center to Fig. 15b.

Reduction (Fig. 16) which means to remove a "perfect tip" [15], i.e. a tree branch, which is a tip in any 2-isomorphic graph. The reduction with respect to row 1 is obtained as follows: add or subtract pairs of those columns (6 and 9) having a non-zero element in row 1, such that the element of the resulting column has a zero in row 1. Augment the matrix by those of the new columns, which have more than



(a) (b) (c)

Fig. 11 - Reduction of a graph by coalescing and removing a cut set.

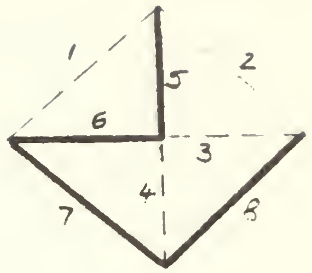
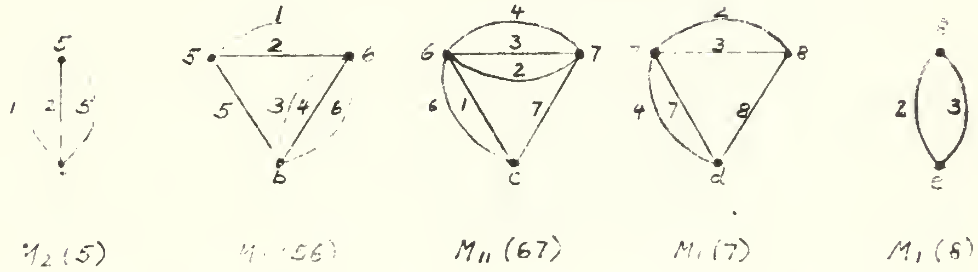
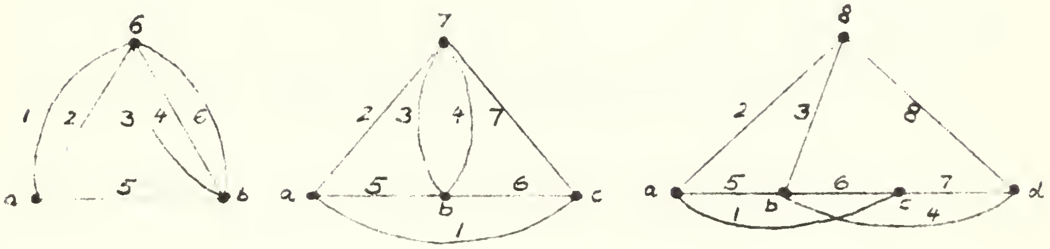


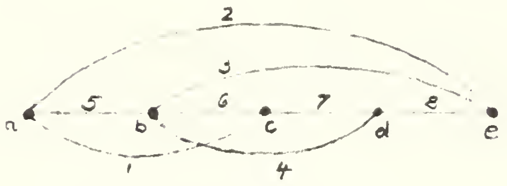
Fig.12 - Graph of example in Section 3.3.



$M_2(5)$ $M_1(56)$ $M_{11}(67)$ $M_1(7)$ $M_1(8)$



combining $M_2(5)$ and $M_1(56)$ and adding $M_1(67)$ and adding $M_1(7)$



finally adding $M_1(8)$

Figure 13

Realization steps of Mayeda's procedure.

one non-zero element and which are not identical with any column already present in the matrix (column a). Finally remove row 1. The reduction procedure attaches to a graph new links (a) corresponding to those columns by which the matrix is augmented. (Each of these links forms a 3 edge loop together with the two links (6 and 9) connected to a perfect tip 1). Removal of row 1 corresponds to removing this perfect tip. (Fig. 16b)

$$-F^t = \begin{array}{c|ccccc} & 6 & 7 & 8 & 9 & a \\ \hline 1 & 1 & 0 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 & 1 & 0 \\ 3 & -1 & -1 & 0 & 0 & 1 \\ 4 & 0 & 1 & -1 & -1 & -1 \\ 5 & 0 & -1 & 1 & 0 & 0 \end{array}$$

Example: Realize $Q = [-F^t \; U]$ by Iri's method. This cut set corresponds to the graph shown in Fig. 17a.

$$-F^t = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 5 & -1 & 1 & 0 & 0 \\ 6 & 1 & -1 & -1 & -1 \\ 7 & 0 & 1 & 1 & 1 \\ 8 & 0 & -1 & -1 & 0 \end{array}$$

No thinning or shrinking are possible. Thus reduce with respect to row 5

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & a \\ \hline 5 & -1 & 1 & 0 & 0 & 0 \\ 6 & 1 & -1 & -1 & -1 & 0 \\ 7 & 0 & 1 & 1 & 1 & 1 \\ 8 & 0 & -1 & -1 & 0 & -1 \end{array}$$

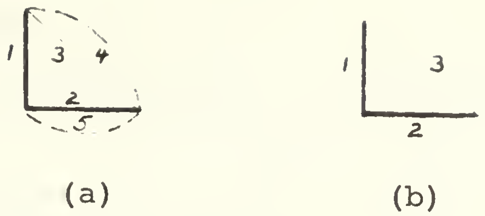


Figure 14
Illustration of "Thinning"

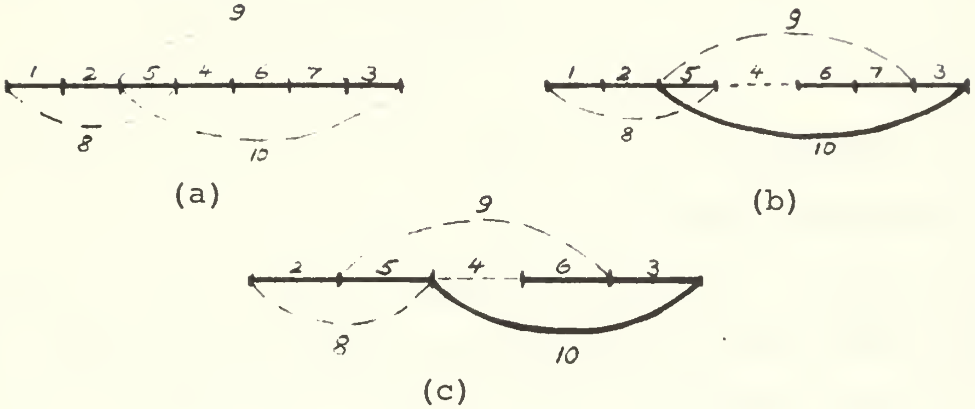


Figure 15

Illustration of "Shrinking"

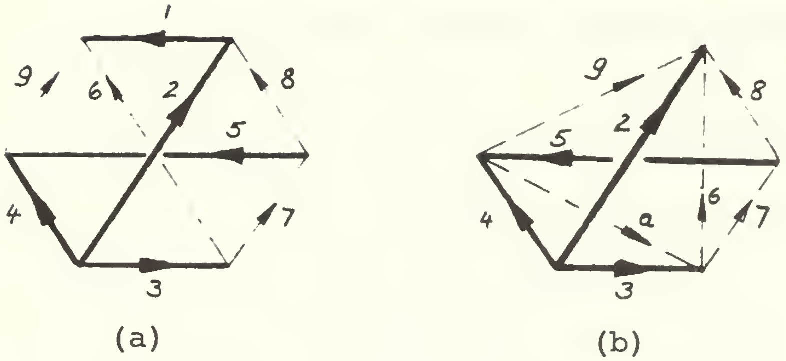


Figure 16

Illustration of "Reduction"

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & a \\ \hline 6 & 1 & -1 & -1 & -1 & 0 \\ 7 & 0 & 1 & 1 & 1 & 1 \\ 8 & 0 & -1 & -1 & 0 & -1 \end{array}$$

Thin with respect to column 1 then augment and reduce with respect to row 6.

$$\begin{array}{c|cccc} & 2 & 3 & 4 & a \\ \hline 7 & 1 & 1 & 1 & 1 \\ 8 & -1 & -1 & 0 & -1 \end{array}$$

Thin with respect to column 4, then remove all but column 2.

$$\begin{array}{c|c} & 2 \\ \hline 7 & 1 \\ 8 & -1 \end{array}$$

Shrink with respect to rows 7 and 8.

The result is a 0 x 1 "void" [14] matrix, representing a single-edge loop.*

The graph is realized starting with the single-edge loop. The links and tree branches are added to the graph, which have been removed by the operations performed on the cut set matrix (Figs. 17b to 17h).

This procedure rests heavily on theorems whose proof is in some cases "of considerable length and tedious" [15].

3.5 Combining Loops within 2-isomorphisms

Two concepts have to be introduced before the methods of Fu [16] and Löfgren [17] can be discussed.

1. In a switching network each branch corresponds to a switch, which is closed (1) or open (0). Connecting two nodes by parallel paths generates a switching function. If one switch within a path

---

* A single-edge loop is formed by coalescing the two nodes of an edge.

is open, the whole path is open. If one path is "closed", the two nodes A and B are connected (Fig. 18).

2. Two graphs G_1 and G_2 are 2-isomorphic if G_2 can be found from G_1 by following operations (Fig. 19a): Cut the graph G_1 at two nodes (n and m) into disjoint subgraphs. Turn one of them around and join them again, yielding G_2. Two graphs are 1-isomorphic if they are cut or joined at one node (Fig. 19b).

Fu's method is based on the following observation: Short circuiting a tree branch j (deleting column j in F), yields a new graph G_j, which has the same loops as the original graph G, except for those loops, which contain the short circuited branch j. In Figs. 20a and 20c the tree branch 2 is short circuited. If the resulting graph G_2 is non-separated (Fig. 20b), the loops which contained branch 2 in the original graph will not contain branch 2 in G_2. If G_2 is separable (Fig. 20d), the loops which contained branch 2 in the original graph will be single-edge loops. The underlying idea of Löfgren's method is that tree branches within each loops have to form an uninterrupted linear path and that the order of tree branches common to more than one loop has to be the same within each loop. During the realization procedure those branches common to two loops are rearranged within 2-isomorphisms. This is illustrated in Fig. 21. Two loops (I and II) shall be combined. Three

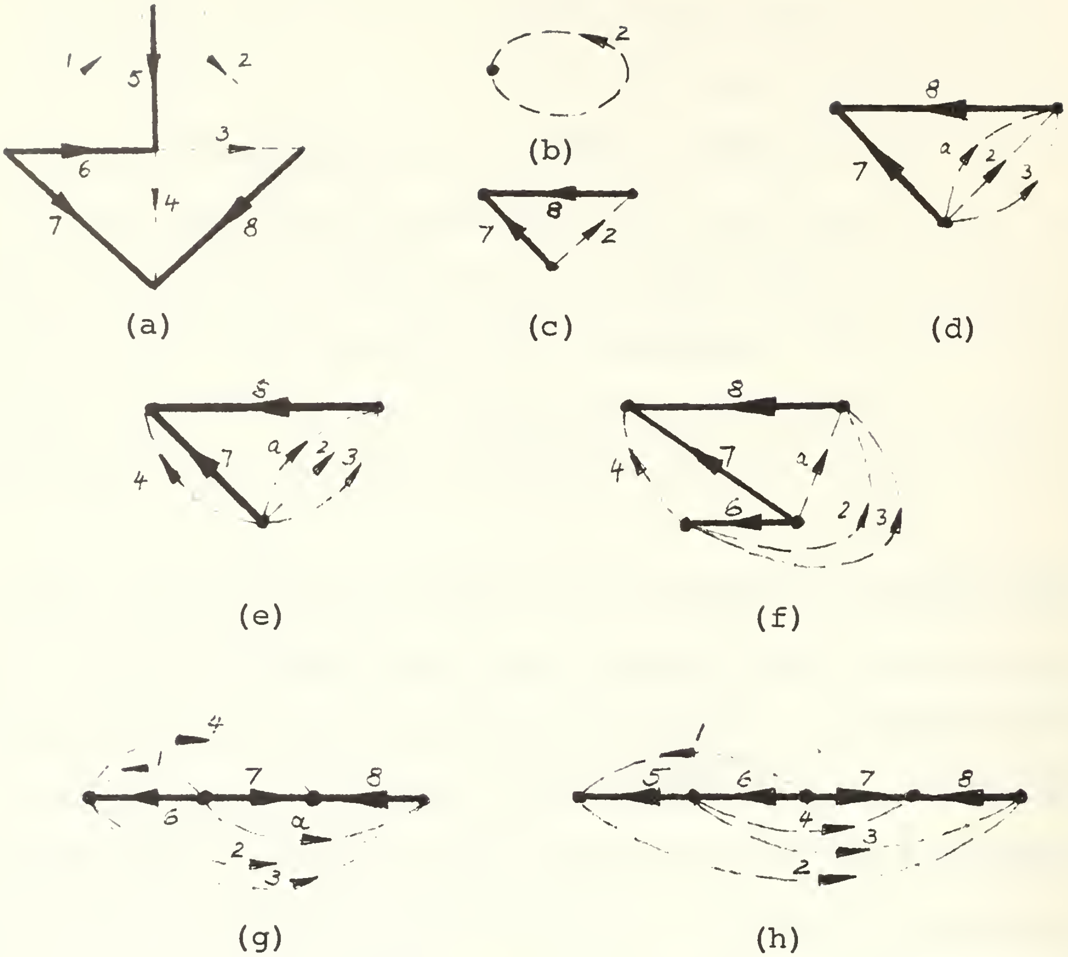


Figure 17
Realization steps in Iri's procedure.

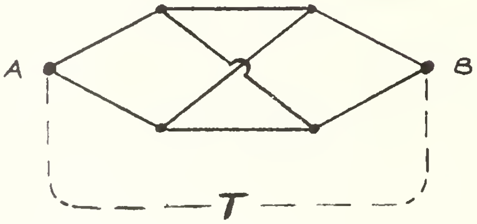


Figure 18
Switching network

branches 1 and 3 do not form an uninterrupted path in loop I. Therefore a 2-isomorphic form of loop I is constructed (Fig. 21c), in which branches 1 and 3 form a linear path before the loops are joined (Fig. 21d).

The realization steps in Fu's method are:

(a) Short circuit all but one of the tree branches and draw the corresponding graph.

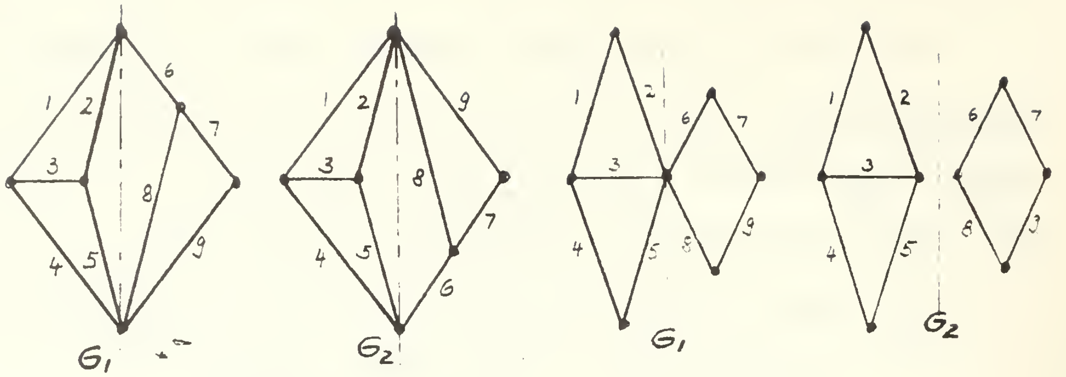
(b) Successively insert the other tree branches in the graph following the conditions demonstrated in Fig. 20.

Although this method looks simple it is not systematic because a tree branch can often be inserted in more than one place. This yields alternate subgraphs, some of which will not lead to a valid graph. These "dead graphs" [16] have to be carried along to ensure that a valid graph will result.

The realization steps of Löfgren's method are:

(a) Choose the loop which contains the smallest number of tree branches and draw the corresponding graph.

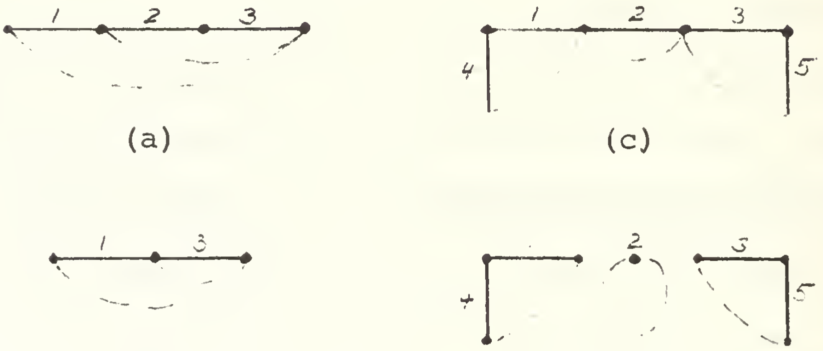
(b) Expand this graph by those loops which share the smallest number of tree branches with the already existing graph. If these common branches do not form an uninterrupted linear path, rearrange them within 2-isomorphisms.



(a) (b)

Figure 19

2-isomorphic (a) and 1-isomorphic graphs (b).

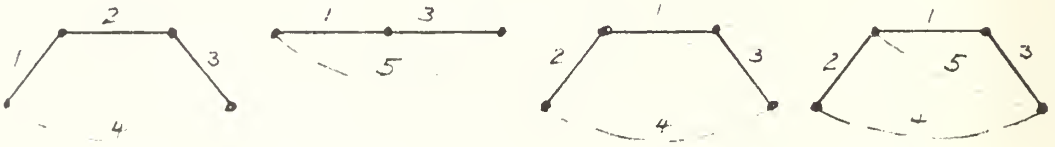


(a) (c)

(b) (d)

Figure 20

Non separated (b) and separable graph (d) resulting from removal of branch from original graphs (a and c).



(a) (b) (c) (d)

Figure 21

Combining loops (a,b) within 2-isomorphisms (c) to a graph (d).

Example: Realize B using Fu's and Löfgren's methods. The given loop matrix corresponds to the graph in Fig. 12.

$$B = \begin{array}{c|cccc:cccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 3 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 4 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{array}$$

The realization steps are shown in Fig. 22.

3.6 Generating the Tree from Path Sets

This method, due to Gould [18], treats the tree branches of each loop as a path set. The tree branches common to different loops form additional path sets. Ordering maximal path sets, i.e. sets, which are not contained in any other path set, subtrees are obtained. This ordering procedure is done by using those path sets which are contained in the respective maximal sets. This procedure is extremely laborious if the tree is not a linear path.

Example: Demonstrate Gould's procedure. The loop set matrix corresponds to Fig. 23a. The graphical representation of the maximal path sets and the forming of the tree is given in Fig. 23b.

$$B = \begin{array}{c|cccccccc} & 1 & 2 & 4 & 8 & 3 & 5 & 6 & 7 & \\ \hline 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & * \\ 2 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & * \\ 4 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & \\ 8 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & * \end{array}$$

set 1 and 2: (3(5)6)

set 4 and 8: (3(67))

*denote
maximal
path sets

These sets are combined using "overlap rules" [18], which merely state that one should combine the subtrees by overlapping corresponding branches to obtain the whole tree.

3.7 Summary

The basic principles of existing procedures have been classified and outlined with the most efficient method of each classification used to solve a simple realization problem. Each of these methods, however, suffers from at least one of the following deficiencies in its application as an engineering tool.

1. The method fails in some cases (Guillemin).
2. The underlying mathematical principles are too abstract for practical applications and thus do not give much insight in the realization procedure (Tutte, Iri).
3. Except for Iri's method, none of these procedures has been put into the form of an algorithm, which can be directly used on a digital computer.

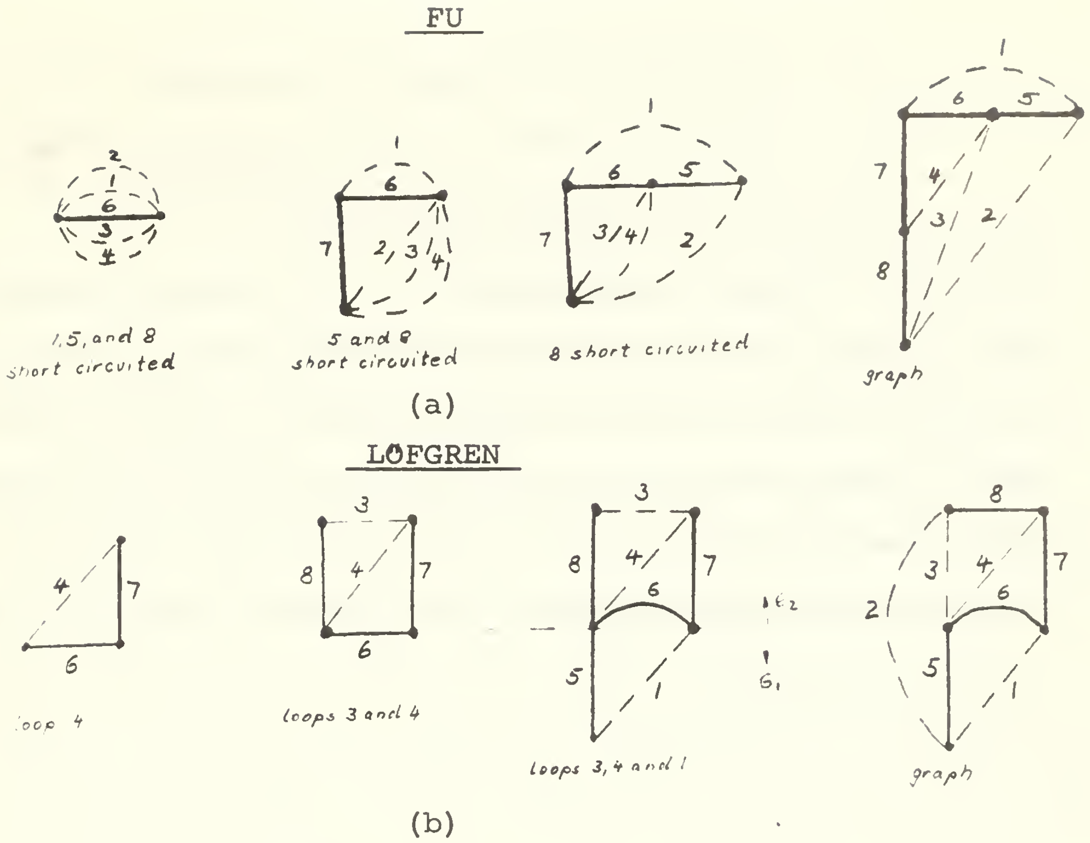


Figure 22

Realization steps in Fu's (a) and Löfgren's method (b).

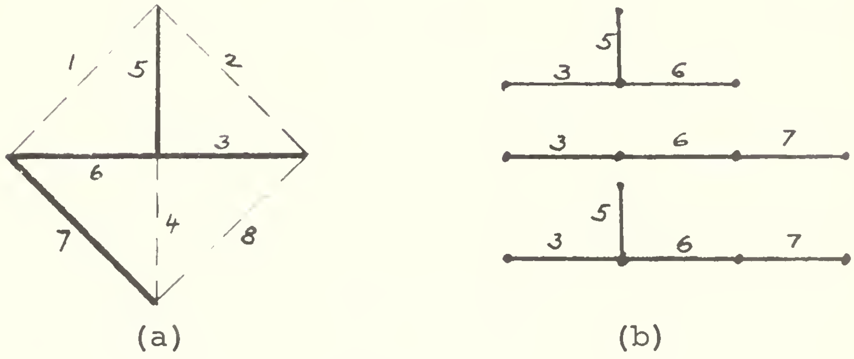


Figure 23

Demonstration of Gould's procedure

4. The method requires too much work particularly as the matrices become more complicated. This is true for all procedures reviewed.

The last statement is especially true for those methods which reveal alternatives while the realization progresses, since only some of these alternatives will yield a valid graph. Thus one has to carry along "alternative" subgraphs, which finally may end up as being a "dead graph" [16]. Moreover, rewriting matrices and redrawing graphs is always a source of error.

REALIZATION OF THE LOOP MATRIX B USING THE COMBINATORIAL APPROACH

4.1 Introduction

The method presented in this thesis takes a combinatorial approach, i.e. joining ℓ loops together such that common branches appear in uninterrupted chains. It thus belongs to the classification of "Generating the Tree from Path-Sets". This method identifies trunk branches, main branches, and limbs, and introduces the concept of "Unique connections". This yields a procedure which is easy to apply and generally more efficient than those reviewed above. The required theorems to develop the procedure are proven rigorously using the method of contradiction. Also the method has been programmed for automatic computation on a digital computer as discussed in Chapter 5.

4.2 The Concept of Trunk Branches, Main Branches, and Limbs

The following information about a graph associated with a given loop matrix can be obtained immediately from the B-matrix:

(a) The number of loops in the graph, and the tree branches plus the defining link for each loop.

(b) Those tree branches which are common to more than one loop.

To extract more information five theorems are stated.

Theorem 4: In each loop the k tree branches form a linear path whose end nodes are connected by a link.

Proof: Assuming there existed a loop whose tree branches do not form a linear path. At least one tree branch would share a node with two branches in a linear path. This would contradict the definition of a loop requiring that exactly two edges have to be incident with each node.

Property 1: If k tree branches belong to only one loop, they may appear within the linear path in any order.

This property holds because permuting any two tree branches does not violate the definition of a loop.

Theorem 5: Two or more tree branches common to more than one loop have to appear in the same sequence in all loops.

Proof: Assuming in Fig. 24 tree branches (b,c,d) form a linear path b-c-d in loop i and b-d-c in loop j. This is possible only if c and d are identical, i.e. tree branches b and c form a linear path in both loops.

As a result of Property 1 and Theorem 5 only those tree branches common to more than one loop will put restrictions on the ordering of tree branches within loops. Those branches that appear in only one loop (their corresponding column in the B-matrix has only one non-zero element), can be placed in series with the link. We will call them limb branches, or just limbs* (branch 9 in Fig. 25). The loop matrix, B of the graph in Fig. 25 is

$$B = \begin{array}{c|ccc|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \hline 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 3 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ & \multicolumn{3}{c}{\text{links}} & \multicolumn{6}{c}{\text{tree branches}} \end{array}$$

That loop which contains the maximum number of tree branches, which are not limbs, shall be called main loop* (loop 1 in Fig. 25). These tree branches (branches 4,5,6,7) form the trunk* of the graph, and are called trunk branches.*

Theorem 6: Trunk branches form a linear path not interrupted by main branches or limbs.

Proof: If there were a limb within the trunk, it could always be moved to one of the two end nodes since it appears only in one loop. If there were a main branch present then, by definition, it should have been included in the trunk.

4.3 Ordering Trunk Branches and the Concept of Unique Connections*

By Theorem 5 each group of trunk branches, belonging to one loop, also appears in the trunk, not interrupted by any other branch. These groups shall be called *chains of trunk* branches or just *chains*. A main branch, together with some of the trunk branches, form linear paths in at least two different loops. These groups of trunk branches also form chains and must share a common node (cf. Theorem 6) to which the main branch is connected (Fig. 25, node a). Thus the following property is obtained:

---

* These definitions are original. The term "limb" appears sometimes in the literature but with a different meaning.

Property 2: The chain and the common node requirements are the only restrictions on the ordering of trunk branches.

Once the trunk is established by combining the chains, starting with the smallest ones, the common node requirements can again be used to attach the main branches to the trunk.

To ensure an efficient connection procedure, those main branches are attached first whose position relative to the trunk is completely specified, i.e. no other branch could take this location without causing an unrealizable graph.

Theorem 7a: A main branch m is *uniquely connected* to the common node between adjacent trunk branches a and b, if either a or b is in each loop in which m is present, provided m appears at least once with a or b, respectively.

Proof: Assume m forms a linear path with trunk chains C1 and C2 in loops L1 and L2. If C1 and C2 would not share a common node, they must be separated by at least one trunk branch. Since m forms a linear path with C1 and C2, respectively, m has to be connected to an end node of C1 and C2, thus forming a loop (Fig. 26a), which is not possible, since trunk and main branches are part of a tree.

Corollary 7: If more than one main branch is uniquely connected to the same node and there exists a loop in which only one of these branches appears, this branch should be connected first. If there is no loop containing all of these main branches, indicating disjoint sets of main branches, one branch of each set will be connected to the common node.

Theorem 7b: A main branch m is uniquely connected to the end node of a chain, if the trunk branch t_1 incident with the end node appears in each loop in which m is present, provided the trunk branch t_2 adjacent to t_1 appears in one but not all of these loops.

Proof: Assume m were connected between t_1 and t_2, then m, t_1 and t_2 could not appear in the same loop. Assume m were connected to the node of t_2, at which t_1 is not incident, then there exists a loop, in which m has to be parallel to t_2 (Fig. 26b). This is impossible, since m, t_1, and t_2 are part of a tree.

Theorem 7b will also be used to attach main branches to those already connected to the trunk.

Once all the unique connections have been carried out, main branches are added to the existing tree structure by using Theorem 4, i.e. they have to form linear paths with the branches already connected.

Should the graph, corresponding to the loop matrix B, be separable, this will be indicated by the following property.

Property 3: If there exist main branches or limbs which do not appear in any loop together with trunk branches, they belong to separate parts of the graph, none of which contains trunk branch.

In case there should be no main branches and/or limbs, the realization procedure will not be affected (see examples in sections 4.5.1 and 4.5.2).

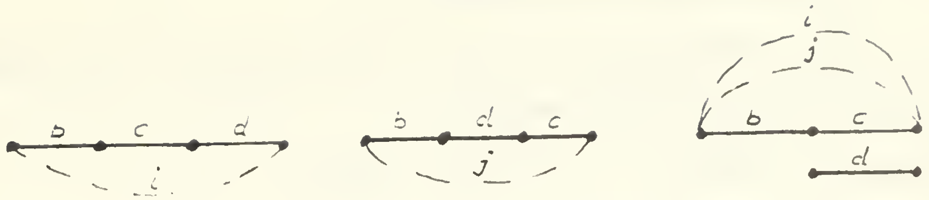


Figure 24

Illustration of Theorem 5.

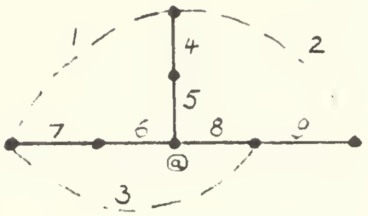


Figure 25

Trunk branches (4,5,6,7), main branches (8), and limbs (9).

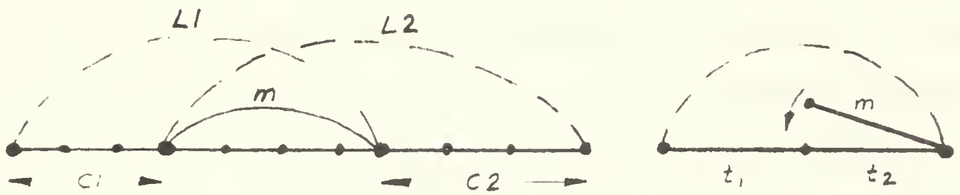


Figure 26

Illustration of Theorems 7a and 7b.

4.4 An Illustrative Example

The procedure shall be demonstrated by working out a realization in full detail.

Given the matrix $B = [U \mid F]$, where:

$$F = \begin{array}{c} \uparrow \\ \text{Links} \\ \text{(Loops)} \\ \downarrow \end{array} \begin{array}{c|ccccccccccccc} & \multicolumn{13}{c}{\longleftarrow \text{Tree Branches} \longrightarrow} \\ & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\ \hline 14 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 15 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 16 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 17 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 18 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 19 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 20 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 21 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 22 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 23 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 24 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{array} \tag{1}$$

Since the procedure does not involve the assignment of directions to the branches, the F matrix, as given, is non-oriented. Should an oriented graph be required the appropriate arrows may be placed on the branches afterwards to correspond to the + or - directions associated with each unity element in the given circuit matrix.

4.4.1 Partitioning the F Matrix

(a) From F form a submatrix, F_ℓ, of those columns which have only one non-zero element. This separates the limbs. The ordering of columns within F_ℓ is not important. The submatrix containing the remaining columns is designated by F_{tm}. Thus $F = [F_{tm} \vdots F_{\ell}]$.

For the given example the limbs correspond to tree branches 10, 12, and 5 and F is partitioned as follows:

$$F = \begin{array}{c|cccccccccc|ccc}
 & 1 & 2 & 3 & 4 & 6 & 7 & 8 & 9 & 13 & 11 & 10 & 12 & 5 \\
\hline
14 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
15 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
16 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
17 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
18 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
19 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
20 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
21 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
22 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
23 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
24 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0
\end{array}$$

$$\longleftarrow F_{tm} \longrightarrow \qquad \longleftarrow F_{\ell} \longrightarrow$$

(b) Find the row of F_{tm} which contains the maximum number of non-zero elements. Separate these columns and form a submatrix designated by F_t. The branches corresponding to these columns are the trunk branches. The ordering of columns F_t is not important. The remaining columns of F_{tm} form the submatrix F_m. The branches corresponding to these columns within F_m is not important. Partition F as follows

$$F = [F_t \vdots F_m \vdots F_{\ell}]$$

For the example, the row corresponding to loop 21 is seen to contain five branches of F_{tm}, namely 1,2,4,9, and 13; which are now identified as trunk branches. In those cases where more than one row contains the maximum number of non-zero elements, alternate selections of trunk branches are possible. For the example, F, is now partitioned as follows:

$$
F = \begin{array}{c|ccccc|ccccc|ccc}
 & 1 & 2 & 4 & 9 & 13 & 3 & 6 & 7 & 8 & 11 & 10 & 12 & 5 \\
\hline
14 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
15 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
17 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
18 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
19 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
20 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
21 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
22 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
23 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
24 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\multicolumn{1}{c}{} & \multicolumn{5}{c}{\longleftarrow F_t \longrightarrow} & \multicolumn{5}{c}{\longleftarrow F_m \longrightarrow} & \multicolumn{3}{c}{\longleftarrow F_\ell \longrightarrow}
\end{array}
$$

4.4.2 Establishing the Trunk

From the F_t matrix list all of the trunk branches which are contained in each row, for those rows that contain more than one trunk branch. Thus for the given example

Loop	Trunk Branches
14 and 24	2-9
21	1-2-4-9-13
23	4-13

Table 1

From F_m, for each main branch, list those trunk branches which appear in the same row. When the main branch is contained in more than one row write these trunk branches on the same line and separate these numbers from the others by a comma. For the given example this listing becomes:

Main Branch	Trunk Branch Chains
3	2, 2-9
6	2-9, 1, 2
7	2, 9
8	1, 2
11	2, 2-9

Table 2

Main branch 6 is connected to trunk branches 2 and 9 in loop 14. Branch 6 is also connected to trunk branch 1 in loop 18, and to trunk branch 2 in loop 19. Commas are used to separate combinations or chains of trunk branches which have the same main branch in common. These chains of trunk branches must have at least one node in common (Theorems 6 and 7). The order in which branches are listed within a chain by itself is not important, since we look at only one loop at a time (Property 1).

Using the second table, starting with rows involving the smallest number of chains of trunk branches, arrange the trunk branches in a sequence which satisfies all of the common node requirements. Thus, for the given example, trunk branch chains 1,2 and 2,9, respectively, must have a common node. This establishes the trunk branch chain 1-2-9. The tabulated requirements that branch 2 and chain 2-9 have common nodes, and that branches 1 and 2 and chain 2-9 have common nodes is also satisfied automatically. Should any of the common node requirements be contradictory the graph would not be realizable.

Inspect this sequence and add additional trunk branches so that all of the trunk branches in Table 1 are included in the trunk chain. This establishes the trunk of the tree. For the example, trunk branches 4 and 13 must be added. Since 4 and 13 must be in a chain it follows that the trunk may be established in four alternate ways as shown in Fig.27, the only restrictions being that chains 1-2-9 and 4-13, respectively, be kept intact.

4.4.3 Making Unique Connections

The information contained in Table 2 is now used to carry out unique connections, applying Theorems 7a, 7b, and Corollary 7.

Main branch 7 is uniquely connected (Theorem 7a) to the node shared by trunk branches 2 and 3.

Main branch 8 is uniquely connected (Theorem 7a) to the node between 1 and 2, but so are main branch 6 (Theorems 7a and 7b), main branch 3 (Theorem 7b), and main branch 11 (Theorem 7b). Applying Corollary 7 the partitioned F-matrix is examined. There is a loop (14) containing only main branch 6 in addition to trunk branches, whereas loops 18 and 19 contain main branches 6 and 8. Thus main branch 6 has to be connected first and is followed by 8.

Main branches 3 and 11 do not appear together with 6 or 8 in any loop, thus they will be part of a different linear path (Corollary 7). Since there is no loop, which besides trunk branches contains only 3 or 11, either one of them may be connected first. Choosing main branch 3 the unique connections are shown in Fig. 28.

4.4.4 Extending Main Branches

Continue the tabulation by listing unconnected main branches and all chains of trunk branches and connected main branches which appear in the same row, using commas to separate chains for different rows. Inspect this part of the table to connect additional main branches to the graph as in section 4.4.3. After these branches are connected extend the table further, continuing until all main branches are connected to the tree.

For the given example the remaining unconnected main branch is 11.

The extended tabulation becomes:

Main Branch	Trunk Branch and Connected Branch Chains
11	2-3-7, 2-3

By Theorem 7b main branch 11 is uniquely connected to main branch 3 at the node where trunk branch 2 is not incident. This completes the connection of main branches.

4.4.5 Connecting the Limbs

Add the limbs to the graph so that they are grouped with the trunk and connected main branches as indicated by the rows in the F matrix.

4.4.6 Connecting the Links

The position of the links relative to the tree are uniquely determined by the B matrix.

The application of the last two steps to the given example yields the final graph as drawn in Fig. 29.

4.5 Further Examples

4.5.1 Graph used in Review of Existing Realization Procedure (No Limbs Present)

$$F = \begin{array}{c|cccc} & 3 & 5 & 6 & 7 \\ \hline 1 & 0 & 1 & 1 & 0 \\ 2 & 1 & 1 & 0 & 0 \\ 4 & 0 & 0 & 1 & 1 \\ 8 & 1 & 0 & 1 & 1 \end{array}$$

The trunk branches can immediately be identified as: 3,6, 7, thus 5 is a main branch.

Table 1	Table 2
6-7	5: 6,3
3-6-7	

Trunk: 7 - 6-3

Since main branch 5 is uniquely connected to the node between trunk branches 6 and 3, the resulting graph is that of Fig. 30.

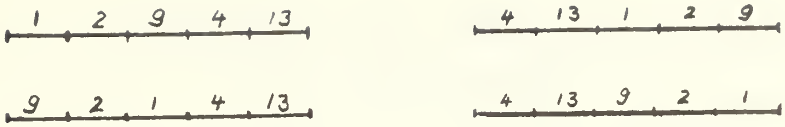


Figure 27
Possible trunks of illustrating example.

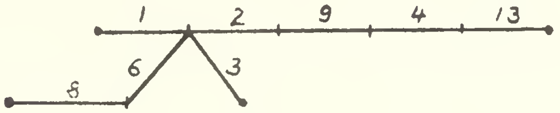


Figure 28
Unique connections of illustrating example.

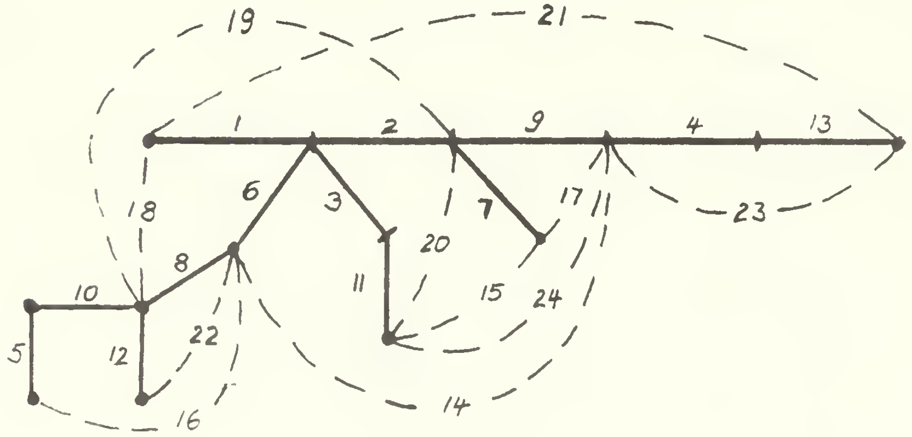


Figure 29
Final graph of illustrating example.

4.5.2 Graph with Separate Parts (No Main Branches Present)

$$F = \begin{array}{c|ccc:cccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline 8 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 9 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 10 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 11 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ & & F_t & & & F_m & & \end{array}$$

Table 1

1 - 2 - 3

2 - 3

1 - 2

Trunk: 1 - 2 - 3

Possible connections for limbs 4 and 5 are shown in Fig.31a. The separate part consists of limbs 6 and 7 and link 11. The final graph is shown in Fig. 31b.

4.5.3 Loop Matrix which is not realizable

This example of an unrealizable loop matrix has been investigated by many authors (for references of [3]).

$$F = \begin{array}{c|ccc} & 5 & 6 & 7 \\ \hline 1 & 1 & 1 & 0 \\ 2 & 1 & 0 & 1 \\ 3 & 0 & 1 & 1 \\ 4 & 1 & 1 & 1 \end{array}$$

Trunk Branches 5,6,7

Table 1

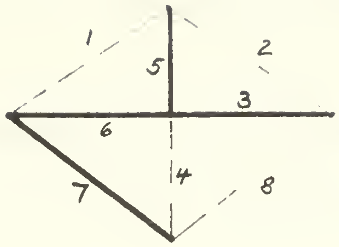
5-6

5-7

6-7

5-6-7

The only way these chains can be combined is shown in Fig. 32. This violates Theorem 6 which states that trunk



(a)

Figure 30

Graph of Section 4.5.1.

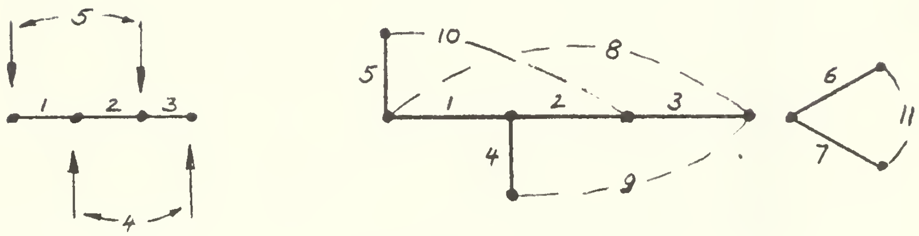
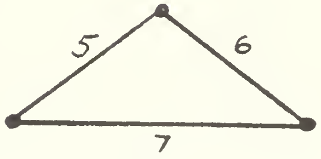


Figure 31

Graph of Section 4.5.2.



(c)

Figure 32

Trunk of Section 4.5.3.

branches have to form a linear path. Moreover Theorem 4 is violated. Thus this loop matrix is not realizable as a graph.

4.5.4 Graph Consisting of Trunk and Main Branches, Limbs, and Separate Parts

Given:

$$F = \begin{array}{c|cccccccccccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
\hline
17 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\
18 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
20 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
21 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
22 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
23 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
24 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
25 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
26 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{array}$$

After partitioning

$$F = \begin{array}{c|cccc|cccccc|cccccc}
 & 1 & 9 & 13 & 16 & 3 & 5 & 6 & 7 & 8 & 12 & 14 & 2 & 4 & 10 & 11 & 15 \\
\hline
17 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
18 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
19 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
20 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
21 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
22 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
23 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
24 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
25 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
26 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
\multicolumn{1}{c}{} & \multicolumn{4}{c}{\longleftarrow F_t \longrightarrow} & \multicolumn{6}{c}{\longleftarrow F_m \longrightarrow} & \multicolumn{6}{c}{\longleftarrow F_\ell \longrightarrow}
\end{array}$$

Table 1:

1-9-13-16
1-13

Table 2:

3	16, 9	-	-	-
5	-	-	6	6-12
6	16	3-16, 3	-	-
7	1, 13-1	1-8, 1-13-8	-	-
8	1, 16, 13-1	-	-	-
12	-	3	6-3, 6	-

The realization steps and the final graph are depicted in Fig. 33.

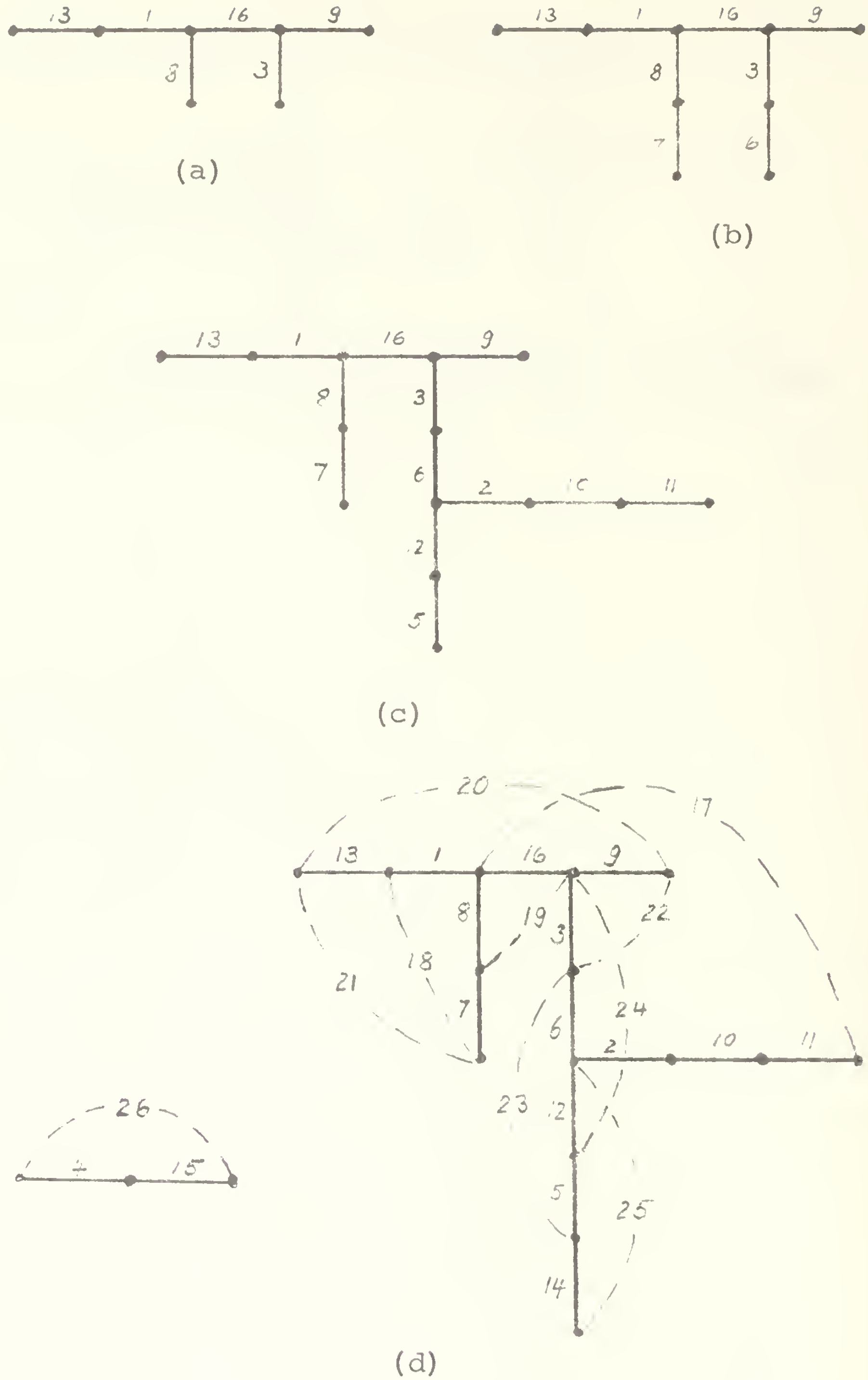


Figure 33

Realization steps and final graph of Section 4.5.4.

PROGRAMMING THE REALIZATION PROCEDURE ON A DIGITAL COMPUTER

5.1 Introduction

In addition to the merits listed in section 3.7 there is another advantage of the procedure presented in this thesis: All realization steps are "final" and a connection will never be nullified. Thus, when this procedure reveals an inconsistancy, the loop matrix is not realizable.

5.2 Input/Output Considerations

The inputs for a computer program have to be a minimum. For this program this is achieved by reading in an *augmented* F matrix, F_a. F is augmented by one row at the top whose elements denote the tree branches corresponding to the columns of F, and by one column at the left side, indicating the links or loops corresponding to the rows of F:

$$F_a = \left[\begin{array}{c|c} 0 & \text{tree branches} \\ \hline \begin{array}{c} \ell \\ i \\ n \\ k \\ s \end{array} & F \end{array}\right]$$

Thus the input for the realization of the B matrix associated with the graph in Fig. 34 is:

$$F_a = \begin{bmatrix} 0 & 5 & 6 & 7 & 8 & 1 & 2 & 3 & 4 \\ 13 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 10 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 9 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 11 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 12 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 14 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The output or print out imposes more problems. There are essentially three possibilities:

1. Graphical output, i.e. have the computer draw the graph.
2. Numerical output, i.e. using a number code or the incidence matrix.
3. Language print out, i.e. giving instructions how to draw the graph.

Possibility 1 requires the use of an additional difficult output program. Possibility 2 requires the use of an efficient coding system or the generation of the incidence matrix.

In both cases information about unique connections is lost. To use a language print out, however, is quite logical for this procedure, because the graph can be drawn in steps once: the possible trunks are listed in their proper sequence; the nodes to which main branches are uniquely connected are given; similarly the nodes to which the remaining main branches and limbs are connected are listed; finally, the two nodes to which each link is connected are stated.

5.3 Partitioning the F Matrix and Establishing the Trunk

The element of each column of the submatrix F within F_a are summed. If the sum equals one, the column corresponds to a limb, and the column (9) of F_a is stored in an array, LIMB. These columns form F_ℓ. Then the elements of each row in F are summed, except those elements which appear in a column identified as a limb. The row (13) having the largest sum is identified as the main loop. Those columns (2,3,4,5) having a non-zero element in this row correspond to trunk branches and are stored in the array, TRUNBR. These columns form F_t. Those columns (6,7,8) having a zero in this row correspond to main branches and are stored in the array MAINBR. These columns form F_m. If there are more than one row with the same maximum sum, the uppermost row of them is identified as the main loop; alternate selections will yield the same graph. This completes the partitioning step.

From F_t all chains of trunk branches are stored in the array KCHAIN, employing row 1 of F_a. Thus 5678,67,58 are stored. From F_m and F_t the common node requirements for each main branch are listed in KCHAIN, avoiding duplications. Thus 57 and 67 are stored. The next step is to combine these chains. This combination procedure is somewhat intricate and shall be explained in more detail. Starting with the smallest chains, stored in the array KCHAIN, larger chains are formed by combinations, stored in the array JTRCH. To ensure that the elements are correctly ordered, the following technique is used: If there is no restriction on the order in which the elements of the resulting JTRCH have to appear, variations are generated such that each element appears at least once as the first or last element. If there is a subsequent KCHAIN, which has at least one element in common with a JTRCH, then there exists one variation of this JTRCH which has this element in an end position. This variation has the correct order, i.e. it appears in this order also in the final trunk or trunks. Since the KCHAIN reflects only a common node requirement, those elements of KCHAIN which are not present in this JTRCH are attached to the latter. The elements by which the JTRCH is augmented are referred to as "non-common" elements.

Ex,:	KCHAIN	38
	JTRCH	1234, 2413
new	JTRCH	24138

Once a JTRCH has been augmented, all variations of it are deleted, i.e. 1234 and 2413. The underlying idea of this technique is that trunk branches form an uninterrupted path (Theorem 6, sect. 4.2). This combination procedure is essentially a problem of pattern recognition which is straight forward for a person to perform. However, the computer has to be programmed to distinguish between 7 combination patterns.

If a KCHAIN (K)

1. cannot be combined with any of the JTRCH, then it forms a new JTRCH (J1). Variations (J2) of this new JTRCH are generated as described above.
   K: 1234; J1: 1234; J2 2143
2. indicates that some of the JTRCH (J) violate a common node requirement, then delete these JTRCH's.

K: 234, J: 2143

3. is completely contained in a JTRCH (J1), then no combinations with K are carried out, but those of the JTRCH's are deleted which violate a common node requirement (J2).

   K: 578, J1: 48572, J2: 84572

4. contains the first (or last) element of a JTRCH (J), then a new JTRCH (JN) is formed, augmenting J by the "non-common" elements of K. All variations of J are deleted.

   K: 7453, J: 1234, JN: 123475

   The variations of JN are 123457, thus each of the new, unordered elements is once in an outer position.

5. contains the first (or the last) element of two different JTRCH's (J1 and J2), then J1 and J2 are combined to form a new JTRCH (JN) with the "non-common" elements in the middle. All variations of J1 and J2 are deleted.

   K: 2615, J1: 36, J2: 57, JN: 362157

   Variation of JN: 361257. If the common elements are both in first position the order of J1 or J2 has to be reversed before the combination.

6. contains the first and the last element of a JTRCH (J), then J is augmented by the "non-common" elements forming a new JTRCH (JN). In the variations the first and the last element of J as well as

each of the "non-common" elements have to be in an end position (6. is a special case of 1.).

K: 36752, J: 572

JN's: 57236, 27563

7. contains the first and the last element of a JTRCH (J1) and at least the first or last element of another JTRCH (J2), then J1, J2, etc., are combined. Each of their end elements and the "non-common" elements have to appear in an end position of one of the variations of the new JTRCH (JN). All variations of J1, J2, etc., are deleted.

   K: 73425, J1: 32, J2: 78

   JN's: 324578, 324587, 432785.

The largest KCHAIN contains all elements of the trunk, because they correspond to the trunk branches present in the main loop (cf. sect. 4.2), we will always be able to generate the complete trunk(s) by this combination procedure.

5.4 Connecting Main Branches and Limbs

When the trunk(s) has been established, the main branches and limbs are connected. In section 4.4 this step of the realization procedure is performed in tabular form. A similar "table" is used in the computer program. For each main branch

1. the number of loops in which it appears is stored in the array MSER,
2. For each of those loops the trunk branches are

stored in the array MBRT and the other main
branches in MBRB.

Example: Tree branch 3 (Fig. 34) is the third main
branch

MSER 3 = 2

MBRT (3,1,1) = 5   MBRB (3,1,1) = 2   (loop 1)

MBRT (3,2,1) = 7   MBRB (3,2,1) = 2   (loop 2)

Unique connections are found by inspecting the above table whether the conditions of Theorems 7a and 7b and Corollary 7 are met by any of the main branches. For this example main branches 3 and 1 are both uniquely connected to the node between trunk branches 5 and 7, either main branch 3 or 1 may be connected first. Main branch 2 is uniquely connected to the outer node of trunk branch 6.

Once all unique connections have been made, a search is performed with the remaining main branches, whether they can be attached to the already connected main branches, according to Theorem 7b. When this step has been performed and there are still unconnected main branches left, they are attached to trunk or already connected main branches such as to satisfy Properties 1 or 3. Limbs are simply connected to the outermost main branch, or leftmost trunk branch if there are no main branches in that specific loop.

5.5 Connecting Links

The links connect the end nodes of the linear paths formed by the tree branches in each loop. In sect. 4.4.6

these end nodes are found by an inspection of the already synthesized tree, represented by a graph, and the rows of the loop matrix. In the computer program an efficient search for the end nodes of the linear paths is achieved by identifying tree branch patterns for each loop. These patterns are demonstrated in Fig. 35, where the capital letters indicate whether trunk branches (T), and/or main branches (M), and/or limbs (L) are present in a loop. The position of the link (dotted line) is indicated as specified by the particular pattern.

This completes the programmed version of the realization procedure. Next, two typical computer solutions are given.

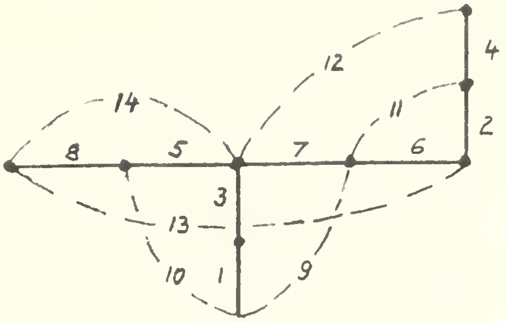


Figure 34

Graph corresponding to loop matrix in Sect. 5.2.

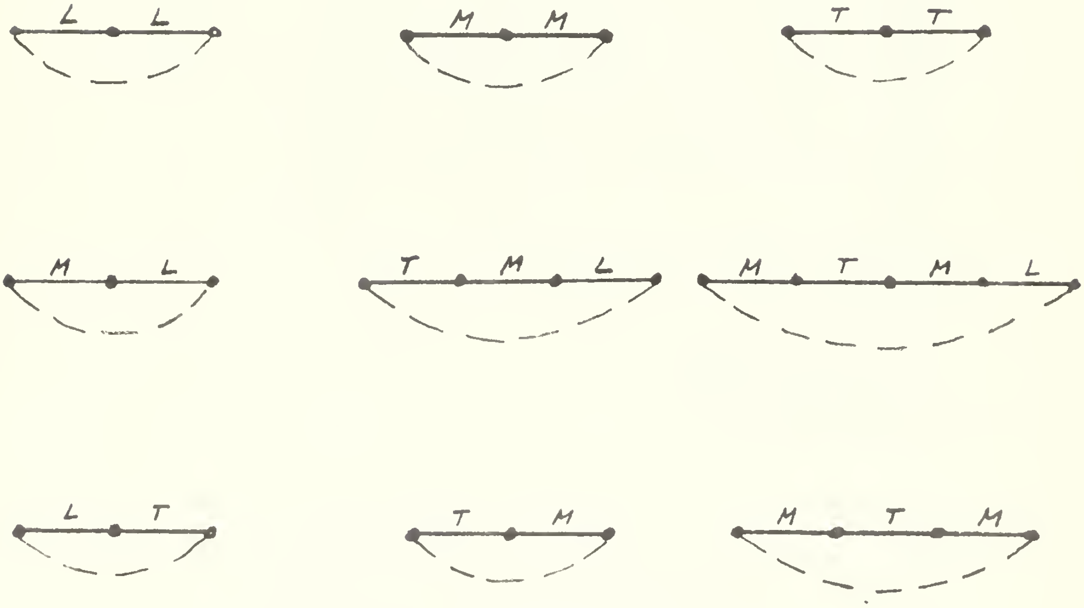


Figure 35

Patterns of link positions with respect to trunk branches (T), main branches (M), and links (L).

5.6 Two Computer Solutions

Computer solution of the example in Section 4.4.

The graph is shown in Figure 29.

```
  F-MATRIX

   0   1   2   3   4   5   6   7   8   9  10  11  12  13
  14   0   1   0   0   0   1   0   0   1   0   0   0   0
  15   0   1   1   0   0   0   1   0   0   0   1   0   0
  16   0   0   0   0   1   0   0   1   0   1   0   0   0
  17   0   0   0   0   0   0   1   0   1   0   0   0   0
  18   1   0   0   0   0   1   0   1   0   0   0   0   0
  19   0   1   0   0   0   1   0   1   0   0   0   0   0
  20   0   1   1   0   0   0   0   0   0   0   1   0   0
  21   1   1   0   1   0   0   0   0   1   0   0   0   1
  22   0   0   0   0   0   0   0   1   0   0   0   1   0
  23   0   0   0   1   0   0   0   0   0   0   0   0   1
  24   0   1   1   0   0   0   0   0   1   0   1   0   0

  F-MATRIX,PARTITIONED

   0   1   2   4   9  13   3   6   7   8  11   5  10  12
  14   0   1   0   1   0   0   1   0   0   0   0   0   0
  15   0   1   0   0   0   1   0   1   0   1   0   0   0
  16   0   0   0   0   0   0   0   0   1   0   1   1   0
  17   0   0   0   1   0   0   0   1   0   0   0   0   0
  18   1   0   0   0   0   0   1   0   1   0   0   0   0
  19   0   1   0   0   0   0   1   0   1   0   0   0   0
  20   0   1   0   0   0   1   0   0   0   1   0   0   0
  21   1   1   1   1   1   0   0   0   0   0   0   0   0
  22   0   0   0   0   0   0   0   0   1   0   0   0   1
  23   0   0   1   0   1   0   0   0   0   0   0   0   0
  24   0   1   0   1   0   1   0   0   0   1   0   0   0

  MAINLOOP IS ROW # 21

5 TRUNKBRANCHES
   1   2   4   9  13

5 MAINBRANCHES
   3   6   7   8  11

3 LIMBS
   5  10  12

  COMMON NODE REQUIREMENTS
   2   9
   4  13
   1   2
   1   2   9
   1   2   4   9  13

2 TRUNKS
   4  13   9   2   1
   9   2   1   4  13
```

```
GRAPH FOR TRUNK 1:
 4 13  9  2  1

BRANCH    SIDE OF        BRANCH

MAIN:

   6      LEFT OF          1

   7      LEFT OF          2

   3      RIGHT OF         2

  11      AND BRANCH#  3 INTERCHANGEABLE

   8      OUTSIDE OF       6

LIMB:

   5      OUTSIDE OF       8

  10      AND BRANCH#  5 INTERCHANGEABLE

  12      OUTSIDE OF       8

LINK:     FROM/TO        BRANCH


  14      OUTSIDE OF       6
          LEFT OF          9

  15      OUTSIDE OF       7
          OUTSIDE OF      11

  16      OUTSIDE OF      10
          INSIDE OF        8

  17      OUTSIDE OF       7
          LEFT OF          9

  18      OUTSIDE OF       8
          RIGHT OF         1

  19      OUTSIDE OF       8
          LEFT OF          2

  20      OUTSIDE OF      11
          LEFT OF          2

  21      LEFT OF          4
          RIGHT OF         1

  22      OUTSIDE OF      12
          INSIDE OF        8

  23      LEFT OF          4
          RIGHT OF        13

  24      OUTSIDE OF      11
          LEFT OF          9
```

A general problem. The graph is shown in Figure 36.

```
  F-MATRIX

   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
  20   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
  21   1   0   0   0   0   0   0   1   1   0   0   0   0   0   0   0   0   0
  22   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
  23   0   1   0   0   0   0   0   1   1   0   0   0   0   0   0   0   0   0
  24   1   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  25   0   0   0   1   0   0   0   0   0   0   0   0   0   0   1   0   0   0
  26   0   0   0   0   1   0   0   0   0   0   0   1   1   0   0   0   0   0
  27   0   0   0   0   1   0   0   0   0   0   0   1   0   0   0   0   0   0
  28   0   0   0   0   1   1   1   0   0   0   0   0   0   0   0   0   0   0
  29   0   0   0   0   0   1   0   0   0   0   0   1   1   1   0   0   0   0
  30   0   0   0   0   0   1   0   0   0   0   0   1   0   0   0   0   0   0
  31   0   0   0   0   0   1   1   0   0   1   1   0   0   0   0   0   0   0
  32   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1
  33   1   1   1   1   1   1   1   0   0   0   0   0   0   0   0   0   0   0
  34   1   1   1   1   1   1   1   0   0   0   0   0   0   0   0   0   0   0
  35   0   0   0   1   0   0   0   0   0   0   0   0   0   0   1   0   0   0

  F-MATRIX,PARTITIONED

   0   1   2   3   4   5   6   7   8   9  12  13  15  16  10  11  14  17  18
  20   1   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
  21   1   0   0   0   0   0   0   1   1   0   0   0   0   0   0   0   0   0
  22   1   1   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0
  23   0   1   0   0   0   0   0   1   1   0   0   0   0   0   0   0   0   0
  24   1   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  25   0   0   0   1   0   0   0   0   0   0   0   1   0   0   0   0   0   0
  26   0   0   0   0   1   0   0   0   0   1   1   0   0   0   0   0   0   0
  27   0   0   0   0   1   0   0   0   0   1   0   0   0   0   0   0   0   0
  28   0   0   0   0   1   1   1   0   0   0   0   0   0   0   0   0   0   0
  29   0   0   0   0   0   1   0   0   0   1   1   0   0   0   0   1   0   0
  30   0   0   0   0   0   1   0   0   0   1   0   0   0   0   0   0   0   0
  31   0   0   0   0   0   1   1   0   0   0   0   0   0   1   1   0   0   0
  32   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1
  33   1   1   1   1   1   1   1   0   0   0   0   0   0   0   0   0   0   0
  34   1   1   1   1   1   1   1   0   0   0   0   0   0   0   0   0   0   0
  35   0   0   0   1   0   0   0   0   0   0   0   1   0   0   0   0   0   0

  MAINLOOP IS ROW # 33

7 TRUNKBRANCHES
   1  2  3  4  5  6  7

6 MAINBRANCHES
   8  9 12 13 15 16

5 LIMBS
  10 11 14 17 18

  COMMON NODE REQUIREMENTS
   1  2
   6  7
   5  6
   1  2  3
   5  6  7
   1  2  3  4  5  6  7
```

```
3 TRUNKS
   7  6  5  1  2  3  4
   1  2  3  4  7  6  5
   4  7  6  5  1  2  3

  GRAPH FOR TRUNK 1:
   7  6  5  1  2  3  4

  BRANCH    SIDE OF         BRANCH

  MAIN:

     12        LEFT OF             5

     16        LEFT OF             1

      8        LEFT OF             2

      9        AND BRANCH#   8  INTERCHANGEABLE

     13        OUTSIDE OF         12

     15        LEFT OF             4       (NOT UNIQUE)

  LIMB:

     14        OUTSIDE OF         13

     10        LEFT OF             7       (NOT UNIQUE)

     11        AND BRANCH#  10  INTERCHANGEABLE

     17        STARTS A SEPARATE PART

     18        AND BRANCH#  17  INTERCHANGEABLE

  LINK:        FROM/TO          BRANCH


     20        OUTSIDE OF         16       28        LEFT OF             7
               RIGHT OF            1                 RIGHT OF            5

     21        OUTSIDE OF          9       29        OUTSIDE OF         14
               LEFT OF             1                 LEFT OF             6

     22        OUTSIDE OF         16       30        OUTSIDE OF         12
               RIGHT OF            2                 LEFT OF             6

     23        OUTSIDE OF          9       31        OUTSIDE OF         11
               RIGHT OF            2                 RIGHT OF            6

     24        LEFT OF             1       32        OUTSIDE OF         18
               RIGHT OF            3                 OUTSIDE OF         17

     25        OUTSIDE OF         15       33        LEFT OF             7
               RIGHT OF            4                 RIGHT OF            4

     26        OUTSIDE OF         13       34        LEFT OF             7
               RIGHT OF            5                 RIGHT OF            4

     27        OUTSIDE OF         12       35        OUTSIDE OF         15
               RIGHT OF            5                 RIGHT OF            4
```

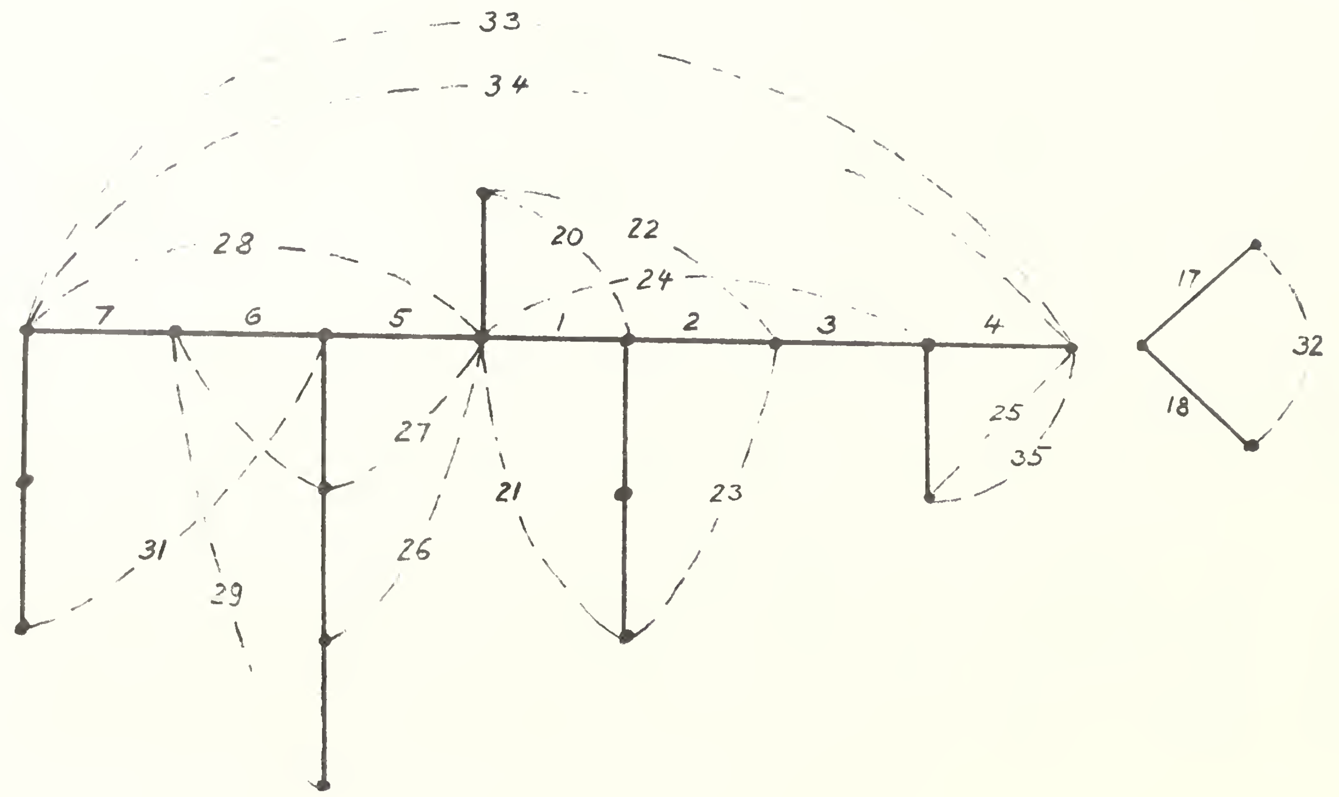


Figure 36

Graph of computer solution 2.

EVALUATION OF THE TOTAL NUMBER OF DIFFERENT GRAPHS HAVING AN IDENTICAL LOOP MATRIX

6.1 Introduction

Up to now this realization procedure has been used to find a graph associated with a given loop set matrix. However, there is another application. It has been pointed out in section 3.1 that there may exist no graph, exactly one graph, or many graphs corresponding to a given loop set matrix. This realization technique gives a way of calculating the total number of different graphs having the same loop set matrix. Although this problem is at the present only of theoretical importance, it demonstrates the generality of the realization procedure presented in this thesis.

6.2 The Concept of Different Graphs

Given two graphs G_1 and G_2 having the same number of nodes and labeled edges. Moreover, they shall have the same number of loops. Each loop in G_1 can be identified with a loop in G_2, i.e. corresponding loops are made up of the same edges.

Definition: Two graphs G_1 and G_2 having the same number of nodes, labeled edges, and identical loops are called different if there is at least one node which is not incident with the same edges in both graphs.

The two graphs in Fig. 37 are different. The concept of different graphs could be useful in integrated circuit layouts. If in Fig. 37 edges 5 and 2 are capacitive and edges 1 and 4 resistive, the circuit arrangement in Fig. 37b could be implemented easier than that in Fig. 37a. To summarize: different graphs are physically different, i.e. the arrangement of their labeled edges relative to each other is not the same. However, these graphs all have the same topological structure.

6.3 Evaluation of the Total Number of Different Trunks

Before calculating the number of different graphs, the number of alternate solution for each realization step is determined. Assuming the trunk is made up of T disjoint chains of trunk branches, i.e. which can be permuted without violating any common node requirement (CNR). Let the ith chain be composed of t_i edges which can be permuted within the chain not violating any CNR. Then the following theorem is true.

Theorem 8: If the trunk is composed of T disjoint chains and each chain has t_i permutable edges, then there exist

$$N_t = (T!/2)\prod_{i=1}^{T} t_i!$$

different trunks.

Proof: This expression is verified using the well known formulae of permutations. The factor one half is introduced since two trunks are not considered to be different if they have the same sequence of branches either from "left to right" or "right to left".

Referring to Fig. 38 the three chains can be permuted in $3!/2$ ways. The branches within each chain can be permuted in $3!$ ways, yielding a total of 648 different trunks.

When there are CNR within the chains, i.e. some branches form subchains, the expression in Theorem 8 has to be modified.

Theorem 9: If the trunk consists of T disjoint chains and the branches in the ith chain form G_1 successively smaller disjoint groups of subchains, each of which having g_{ij} disjoint components, then there exist

$$N_t = (T!/2) \prod_{i=1}^{T} \prod_{j=1}^{G_i} g_{ij}!$$

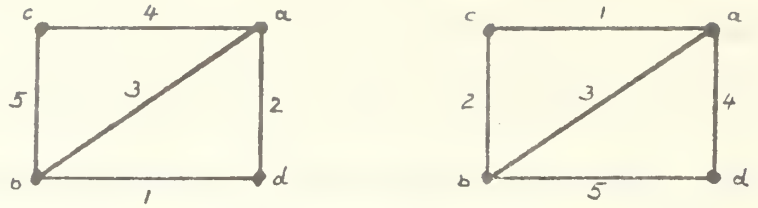
Proof: This theorem is verified applying combinatorial mathematics. In Fig. 39 chains 1 and 2 can be permuted in $(2!/2) = 1$ way. Chain 1 does not have any subchains, thus there is only one CNR, i.e. $G_1 = 1$. The edges 1 and 2 can be permuted in $2!$ ways. Chain 2 consists os two subchains, thus there are 3 CNR within chain 2, i.e. $G_2 = 3$. The product $\prod_{j=1}^{3} g_{2j}!$ is composed of the terms

$g_{21} = 1!$ for the elements in subchain 2a

$g_{22} = 3!$ for the elements in subchain 2b

$g_{23} = 2!$ for the subchains 2a and 2b in chain 2

Thus there exist a total of $(2!/2)\ [2!]\ [1!3!2!] = 24$ different trunks. This completes the calculation of the number of different trunks.



(a) (b)

Figure 37

Different graphs

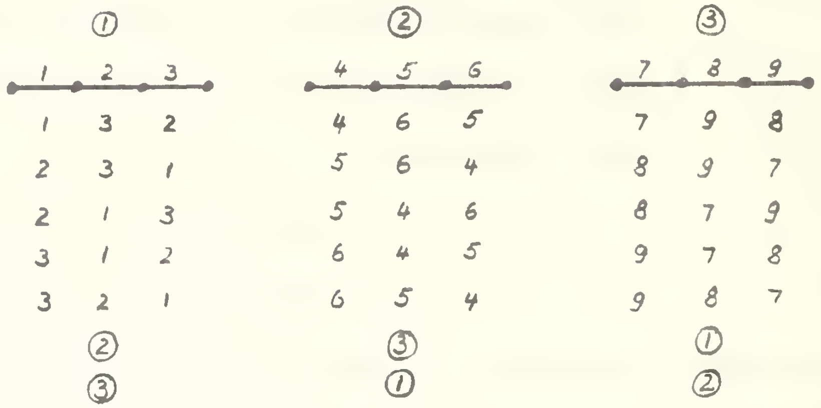


Figure 38

Different trunks composed of chains 1, 2, and 3.

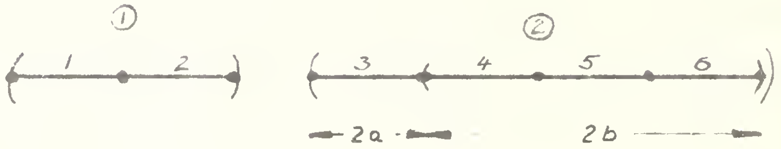


Figure 39

Different trunks.

6.4 Evaluation of the Total Number of Different Main Branch Connections

In the process of connecting main branches to the trunk or already connected main branches, there can arise four possibilities:

1. One main branch is uniquely connected to a node.
2. Two or more main branches are uniquely connected to the same node.
3. One main branch can be connected to either end of a linear path of trunk branches and/or already connected main branches.
4. More than one main branch can be connected as in 3.

Clearly case 1 does not yield alternate solutions. Assuming there are M groups of main branches uniquely connected to the same node, the ith group having m_i elements. Assuming, moreover, there are N groups of main branches that could be connected to either one of the two end nodes of a linear path, the jth group having n_j elements. Then following theorem holds:

Theorem 10: If the main branches can be uniquely connected in M groups to the same node, such that each elements of the group may be connected first, and N groups of main branches can be connected to either of two end nodes of a linear path, then there are

$$N_m = \prod_{i=1}^{M} m_i! \prod_{j=1}^{N} (n_j+1)!$$

different ways to connect these main branches.

Proof: Referring to Fig. 40a the 3 main branches can be connected in 3! different ways to node v. From Fig. 43b it follows that main branch 4 can be connected in two ways to end nodes a or b. Fig. 40c reveals that there are 4x3! different ways to connect the main branches 5,6, and 7 to either one of the end nodes c and d. If all main branches are connected to one node, 3! different connections result. Alternately, one branch can be selected in 3 ways and then connected to one node. The remaining 2 main branches are then attached in 2! different ways to the other node. This again, yields 3! different possibilities. This completes the calculation of the different possibilities to connect the main branches.

6.5 Evaluation of the Total Number of Different Limb and Link Connections

Finally the number of alternate connections for limbs and links is given by

Theorem 11: If there are L loops, the ith loop having ℓ_i limbs, then the limbs and the links can be connected in

$$N_\ell = \prod_{i=1}^{L} (\ell_i+1)!$$

Proof: Referring to Fig. 41 there are 3! ways to connect the three limbs, assuming the link is connected last. However, the link can be placed in four different positions, thus there are a total of 4x3! different ways to connect the three limbs and the link. This completes the number of different ways limbs and links can be connected.

6.6 Evaluation of the Total Number of Different Graphs

By the definition of the term "different graph" each of the alternate connections yields a different graph. Thus the following theorem holds:

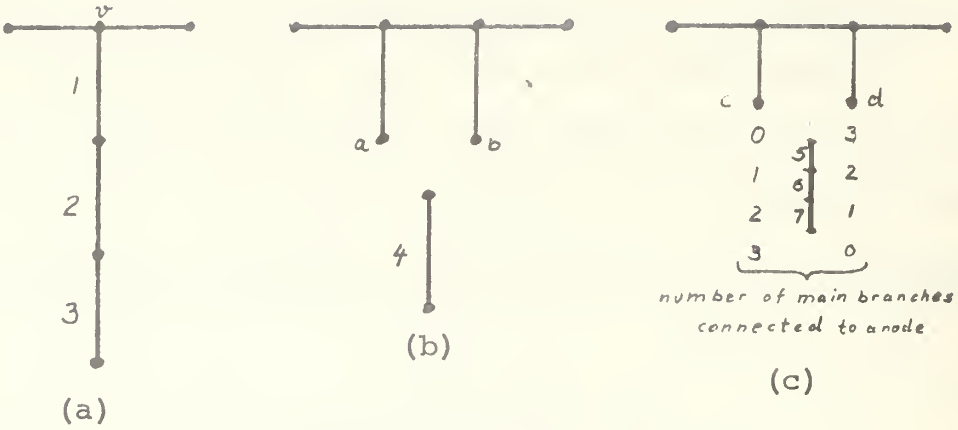
Theorem 12: If the realization procedure presented reveals that

1. The trunk consists of T disjoint chains, the ith chain forming G_i successively smaller disjoint groups of subchains each having g_{ij} disjoint components,
2. the main branches can be uniquely connected in M groups to one node and attached in N groups to either one of two end nodes,
3. there are L loops, the rth loop having ℓ_r limbs then there exist

$$N = \left[(T!/2) \prod_{i=1}^{T} \prod_{j=1}^{G_i} g_{ij}! \right] \left[\prod_{k=1}^{M} m_k! \prod_{p=1}^{N} (n_p+1)! \right] \left[\prod_{r=1}^{L} (\ell_r+1)! \right]$$

different graphs corresponding to a given fundamental loop or cut set matrix.

The unknowns present in this expression can be obtained from the computer program. The common node requirements are listed in the output for this purpose.



(a) (b) (c)

Figure 40

Different main branch connections

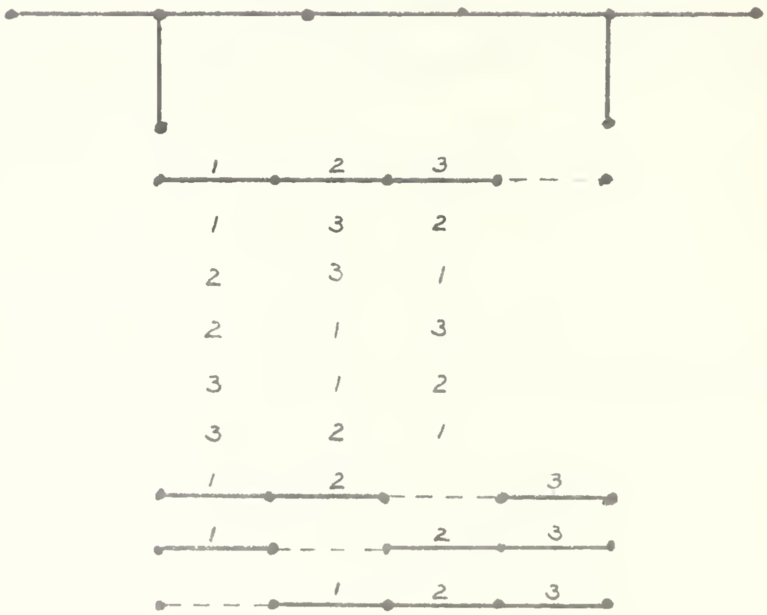


Figure 41

Different limb connections

CONCLUSION AND SUGGESTION FOR FURTHER RESEARCH

The efficiency and generality of a new realization procedure for a fundamental loop or cut set matrix has been demonstrated. The concept of trunk branches, main branches, and limbs proves to be useful in describing graphs. The idea of physically different networks is introduced which should prove to be useful in finding or classifying equivalent networks.

BIBLIOGRAPHY

1. Kuo, F. F., "Network Analysis by Digital Computers", Proc. IEEE, Vol. 54, pp.820-829, June 1966.

2. Seshu, S. and Reed, M. B., "Linear Graphs and Electrical Networks", Addison-Wesley, Reading, Mass., 1961.

3. Kim, W. H. and Chien, R. T., "Topological Analysis and Synthesis of Communication Networks", Columbia University Press, 1962.

4. Desoer, C. A. and Kuh, E. S., "Basic Circuit Theory", McGraw-Hill, New York, 1966.

5. Elgerd, O. I., "Control Systems Theory", McGraw-Hill, New York, 1967.

6. Bashkow, T. R., "The A Matrix, New Network Description", IRE Trans. on Circuit Theory, Vol. CT-4, pp.117-120, Sept., 1957.

7. Kuh, E. S., and Rohrer, R. A., "The State-Variable Approach to Network Analysis", Proc. IEEE, Vol. 53, pp.672-686, July 1965.

8. Timothy, L. K., and Bona, B. E., "State Space Analysis, An Introduction", McGraw-Hill, New York, 1968.

9. Biorci, G. and Civalleri, P. P., "On the Synthesis of N-Port Networks", IRE Trans. On Circuit Theory, Vol. CT-8, pp.22-28, March 1961.

10. Halkias, C. C. and Kim, W. H., "Realization of Fundamental Circuit and Cut-Set-Matrices", IRE International Convention Record, PT 2, pp.8-15, 1962.

11. Tutte, W. T., "An Algorith, for Determining Whether A Given Binary Matroid Is Graphic", Proc. Am. Math. Soc., Vol. 11, No.6, pp.905-917, Dec. 1960.

12. Mayeda, W., "Necessary and Sufficient Conditions for Realizability of Cut-Set-Matrices", IRE Trans. on Circuit Theory, Vol. CT-7, pp.79-81, March 1960.

13. Auslander, L. and Trent, H. M., "Incidence Matrices and Linear Graphs", Journal of Mathem. and Mechanics, Vol.8, No.5, pp.827-835, Sept. 1959.

14. Guillemin, E.A., "How to Grow Your Own Trees from Given Cut-Set or Tie-Set Matrices", IRE Trans. on Circuit Theory, Vol. CT-6, pp.111-126, May 1959.

15. IRI, M., "Application of Graph Theory to Switching Networks - A Fundamental Problem and its Solution", Progress in Radio Science, 1960-1963, Vol. VI, Elsevier Publishing Company, New York 1966.

16. Fu, Y., "Realization of Circuit Matrices", IEEE Trans. on Circuit Theory, Vol. CT-12, No.4, pp.604-607, Dec. 1965.

17. Löfgren, L., "Irredundant and Redundant Boolean Branch Networks", IRE Trans. on Circuit Theory, Vol. CT-6, Special Supplement, pp.158-175, 1959.

17. Gould, R., "Graphs and Vector Spaces", Journal of Math. Physics, Vol.3, pp.193-214, 1958.

18. Parker, S. R. and Lohse, H. J., "A Direct Procedure for the Synthesis of Network Graphs from a Given Fundamental Loop or Cut Set Matrix", IEEE Trans. on Circuit Theory, May 1969.

19. Kirchhoff, G. "Über die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Vertheilung galvanischer Ströme geführt wird." Ann. Phys. und Chem., Vol. 72, No.12, pp.497-508, Dec.1847.

APPENDIX I

Glossary of Topological Terms

The definition of topological terms used in this thesis are summarized below.

Adjacent: Two edges are adjacent, if they have a vertex in common.

Basic: See Fundamental

Branch: See Tree Branch

Connected: A graph is connected if there exists a path between any two nodes; a connected graph cannot contain any isolated node.

Co-tree: The edges of the graph which are not part of the tree form the complement of the tree or co-tree.

Cut Set: A set of edges forms a cut set if the removal of this set "cuts" the graph into separate parts, however, removing all but one of these edges leaves the graph connected.

Edge: Two distinct points (end points) and the line segment joining them form an edge.

Fundamental Loop: A fundamental loop consists of one link and the tree branches connecting its nodes.

Fundamental Cut Set: A cut set which besides links cuts only one tree branch.

Fundamental Loop Set Matrix

$$B = \left[\begin{array}{c|c} \text{links} & \text{tree branches} \\ U & F \end{array}\right] \ \text{loops}$$

$b_{ij} = 1$ if edge j is present in loop i.

Fundamental Cut Set Matrix

$$\begin{array}{cccc} & \text{links} & \text{tree branches} & \\ Q = & \left[-F^t \right. & \left. \vdots \quad U \right] & \text{cut sets} \end{array}$$

q_{ij} = 1 if edge j is cut by the cut i.

Graph: A "linear graph" or just graph is a collection of edges with no single-edge loops.

Incident: A node and an edge are incident if the node is an endpoint of the edge.

Incidence Matrix:

$$\begin{array}{ccc} & \text{edges} & \\ A = & \left[a_{ij} \right] & \text{nodes} \end{array}$$

a_{ij} = 1 if edge j and node i are incident

Incidence Set: The set of edges incident at one node from an incidence set.

Isomorphisms: If a graph G_1 can be broken at a single node into two disjoint subgraphs or two disjoint subgraphs are joined at a single node, then the resulting graph and G_1 are 1-isomorphic. If a graph G_1 is cut at two nodes into two disjoint subgraphs, one of them is turned around and then joined with the other at the two nodes, G_1 and the resulting graphs are 2-isomorphic.

Limb: A tree branch which appears in only one loop.

Linear Graph: See Graph.

Link: An edge of a graph which is not a tree branch, is a link. The links form the co-tree.

Loop: A connected subgraph in which exactly two edges are incident with each node.

Main Branch: A tree branch which appears in more than one loop and is not part of the trunk.

Main Loop: The loop of a graph containing the maximum number of those tree branches which are common to more than one loop.

Node: The endpoint of an edge is called a node or vertex. Isolated points of a graph are called "isolated nodes".

Oriented: If an edge is assigned a direction, the edge is oriented. If a graph is made up of oriented edges, the graph is called oriented.

Path: A linear path or just path, between nodes i and j is a sequence of edges. Each node of the path is shared by exactly two edges except nodes i and j which form the endpoints of the path.

Separable Graph: A connected graph is separable if it contains at least one subgraph which has only one node in common with its complement.

Single-Edge Loop: Coalescing the two nodes of an edge generates a single-edge loop.

Tree: A connected subgraph containing all nodes of a graph but no loops is called a tree of a graph.

Tree branches: The edges of a graph which are part of the tree are called tree branches or just branches.

Trunk: The linear path which is formed by the tree branches appearing in the main loop (omitting the limbs) is called the trunk of the tree.

Trunk Branches: The tree branches which form the trunk are called trunk branches.

Unique Connection: If the position of a main branch relative to the trunk and/or connected main branch is exactly specified, the main branch is uniquely connected with respect to those branches.

Vertex: See Node.

APPENDIX II

Transformation of Current and Voltage Sources

For each electric network there exists an associated linear graph. This graph can be found by

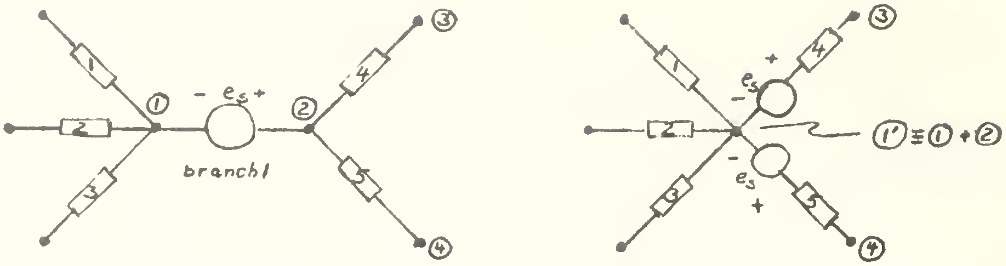
1. open circuiting ideal current sources,
2. short circuiting ideal voltage sources,
3. and replacing passive elements by edges.

If there are branches in the network which consist of ideal sources only, these sources can be transformed such that each of the ideal current sources is parallel to a passive element, and each of the ideal voltage sources is in series with a passive element. This transformation technique is taken from Reference 4. It is assumed that there are no loops formed by independent voltage sources. Assuming this were not the case, one of these sources had to be dependent by KVL, which contradicts the assumption.

Since voltage sources do not form loops, an edge consisting of a voltage alone can be identified as a tree branch (Fig. 42a). Removal of this tree branch cuts the tree into two separate parts. The voltage source is placed in series with each element corresponding to the tree branches in one of the separate parts. The two nodes of the removed edge are then coalesced (Fig. 42b). This transformation does not change the network equations. This is verified by writing KVL for the loops containing the tree branches into which the voltage sources have been inserted, or KCL for the two nodes which have been coalesced.

Since current sources do not form cut sets, and edge consisting of a current source alone can be identified as a link (Fig. 43a). The current source is placed in parallel with each element corresponding to those tree branches which form a loop with that link (Fig. 43b). This transformation does not change the network equations. This is verified by writing KCL for the two nodes of that link, or KVL for the voltage between the two nodes of that link.

Thus each branch of an electric network can be represented in its most general form as in Fig. 44, where the sources (v_{sk} and j_{sk}) may be absent, the passive element R_k is always present.



(a) (b)

Figure 42

Voltage source transformation

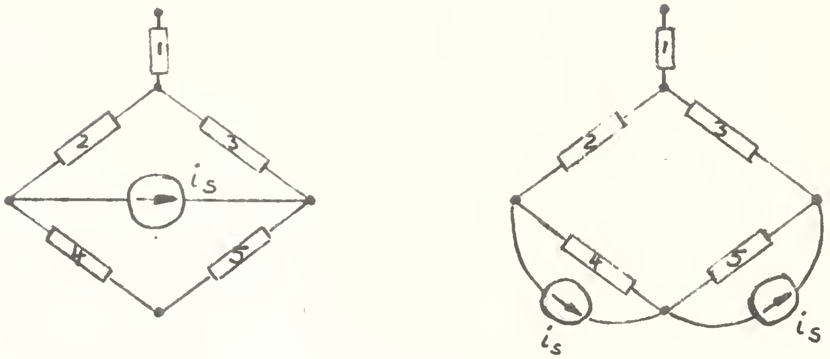


Figure 43

Current source transformation

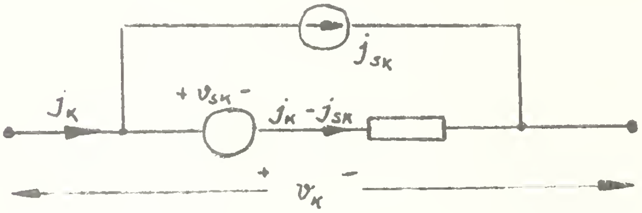


Figure 44

Typical branch of an electric network

APPENDIX III

Computer Program

```
C THIS PROGRAM REALIZES A FUNDAMENTAL LOOP SET MATRIX.
C  INPUT:
C      1) ROWS AND
C      2) COLUMNS OF
C         THE AUGMENTED F-MATRIX,WHOSE ELEMENTS IN COLUMN 1
C         CORRESPOND TO LINKS,AND WHOSE ELEMENTS IN ROW 1
C         CORRESPOND TO TREE BRANCHES.
C      3) THE AUGMENTED F-MATRIX,ONE ROW PER CARD.
C OUTPUT:
C       1) AUMGNENTED F-MATRIX PUT IN,
C       2) PARTITIONED F-MATRIX,
C       3) COMMON NODE REQUIREMENTS,
C       4) NUMBER OF DIFFERENT TRUNKS WHICH ARE ARE IMPORTANT
C          FOR THE ALGORITHM USED IN THIS PROGRAM,
C       5) ONE GRAPH.IF ALTERNATE,HOWEVER EQUIVALENT,
C          REALIZATIONS EXIST,SET THE VARIABLE "NOGRAF" TO
C          TO THE DESIRED NUMBER.IF THE PROGRAM RECOGNIZES
C          FEWER ALTERNATIVES,ONLY THESE ARE PRINTED.
        IMPLICIT INTEGER (A-Z)
      DIMENSION F(30,30),LIMB(20),NOTLIM(20),MAINBR(20)
      DIMENSION TRUNBR(20)
      DIMENSION MEND(30),GRAPH(60,3),TRUNK(20,20),FPART(30)
      DIMENSION MSER(30),MBRB(30,10,10),MBRT(30,10,10)
      DIMENSION CONECT(40)
      DIMENSION MTS(5,4),MT(5),MBGS(5),MAMBIG(9,9)
      DIMENSION MSLET(30,10),MSLEB(30,10),MCOM(5,5),MC(9)
      DIMENSION JTRCH(30,30),LENGTH(20),JLAST(10),JFIRST(10)
      DIMENSION JTEMP1(30)
      DIMENSION JADD(30,9),MUNIQ(9,4),MCOMP(10) ,JINTLE(10)
      DIMENSION KSTORE (20),KCHAIN(20,10,20),KC(10)
      DIMENSION JTEMP(30),JSEQ(30)
      DIMENSION KTRCH(10,10),KLE(20),JSTOR(5,10)
      COMMON /CH/ KCHAIN,KSTORE,KC
      COMMON/PT/JTRCH,LENGTH,JTWO,JLAST,JFIRST,JADD,
     1JTEMP1,JT,JF,JL
      COMMON/LLO/ ROW,COL,F,TR,TRUNBR,M8,TRUNK,MBR,MAINBR,
     1MEND,MGR,GRAPH,L,LIMB,ME
      COMMON /NOD/JDELET(40),JD,COMB(40),C,J
      READ (5,299) ROW
      READ (5,299) COL
  299 FORMAT (I2)
      DO 297 IN=1,ROW
      READ (5,298) (F(IN,I),I=1,COL)
  297 CONTINUE
  298 FORMAT (30I2)
      NOGRAF=1
C PARTITION F MATRIX
C FIND LIMBS
      L=0
      NL=0
      DO 2  I1=2,COL
      COLSUM=0
      DO 3 I2=2,ROW
    3 COLSUM=COLSUM+F(I2,I1)
      IF (COLSUM.GT.1)GOTO 4
      L=L+1
      LIMB(L)=I1
      GOTO 2
    4 NL=NL+1
      NOTLIM(NL)=I1
    2 CONTINUE
C FIND MAINLOOP
      ITEST=0
      DO 5  I3=2,ROW
      ROWSUM=0
      DO 1  I4=1,NL
      COLF=NOTLIM(I4)
```

```
 1    ROWSUM=ROWSUM+F(I3,COLF)
      IF(ROWSUM.LE.ITEST)GOTO 5
      TROW=I3
      ITEST=ROWSUM
 5    CONTINUE
      TR=0
      MBR=0
C FIND TRUNK BRANCHES AND MAIN BRANCHES
      DO 6  I5=1,NL
      COLF=NOTLIM(I5)
      IF (F(TROW,COLF).EQ.1) GOTO 7
      MBR=MBR+1
      MAINBR(MBR)=COLF
      GOTO 6
 7    TR=TR+1
      TRUNBR(TR)=COLF
 6    CONTINUE
      WRITE(6,350)
      DO 301 IK=1,ROW
      WRITE (6,309) (F(IK,I),I=1,COL)
 301  CONTINUE
      FPART(1)=1
      IF(TR.EQ.0)GO TO 302
      DO 303 N50=1,TR
 303  FPART(1+N50)=TRUNBR(N50)
 302  CONTINUE
      IF(MBR.EQ.0) GO TO 304
      DO 305 N51=1,MBR
 305  FPART(1+TR+N51)=MAINBR(N51)
 304  CONTINUE
      IF(L.EQ.0) GO TO 306
      DO 307 N52=1,L
 307  FPART(1+TR+MBR+N52)=LIMB(N52)
 306  CONTINUE
      WRITE(6,351)
      DO 308 N53=1,ROW
      WRITE(6,309)(F(N53,FPART(I)),I=1,COL)
 308  CONTINUE
      WRITE(6,353) F(TROW,1)
      WRITE(6,354) TR
      N54=1+TR
      WRITE(6,309)(F(1,FPART(I)),I=2,N54)
      WRITE(6,355) MBR
      N55=2+TR
      N56=1+TR+MBR
      WRITE(6,309) (F(1,FPART(I)),I=N55,N56)
      WRITE(6,356) L
      N57=2+TR+MBR
      N58=1+TR+MBR+L
      WRITE(6,309) (F(1,FPART(I)),I=N57,N58)
C FIND CHAINS OF TRUNKBRANCHES FROM FT
      DO 8 I6=1,TR
 8    KC(I6)=0
      DO  9  K1=2,ROW
      KSUM=0
      DO 10  K2=1,TR
      COLF=TRUNBR(K2)
      IF (F(K1,COLF).EQ.0)GOTO 10
      KSUM=KSUM+1
      KSTORE(KSUM)=F(1,COLF)
 10   CONTINUE
      IF(KSUM.LE.1) GOTO 9
      CALL CHAINS(KSUM)
 9    CONTINUE
C FIND CHAINS OF TRUNKBRANCHES AS INDICATED
C   BY COMMON NODF REQUIREMENT IN FM
      IF (MBR.EQ.0) GOTO 11
      DO 12  K3=1,MBR
      MAINF=MAINBR(K3)
      MFLAG=0
      MFLAG2=0
      KT=0
      DO 13  K4=2,ROW
      IF(F(K4,MAINF).EQ.0)GOTO 13
      KSUM=0
      DO 14  K5=1,TR
      COLF=TRUNBR(K5)
      IF(F(K4,COLF).EQ.0)GOTO 14
      KSUM=KSUM+1
      KSTORE(KSUM)=F(1,COLF)
 14   CONTINUE
      IF (KSUM -1) 13,15,17
 15   MFLAG=MFLAG +1
      IF (MFLAG.GT.1)GOTO 16
      MTEMP=KSTORE(1)
       GOTO 13
 16   MTEMP1=KSTORE(1)
       IF (MTEMP.EQ.MTEMP1) GOTO 13
      KSUM=2
      KSTORE(1)=MTEMP
      KSTORE(2)=MTEMP1
      CALL CHAINS (KSUM)
      MFLAG=0
      MFLAG2=1
 17   KT=KT+1
      DO 18 K6=1,KSUM
 18   KTRCH(KT,K6)=KSTORE(K6)
      KLE(KT)=KSUM
 13   CONTINUE
      IF (MFLAG.GE.1.AND.MFLAG2.EQ.0) GOTO 19
      GOTO 20
 19   KT=KT+1
      KTRCH(KT,1)=MTEMP
      KLE(KT)=1
 20   CONTINUE
      IF (KT.LE.1) GOTO 12
      KCHECK=KT-1
      DO 21 K7=1,KCHECK
      K8=KLE(K7)
      K9=K7+1
      DO 22 K10=K9,KT
      KSUM=0
      K11=KLE(K10)
      DO 23 K12=1,K11
      KNOT=0
      DO 24 K13=1,K8
      IF (KTRCH(K10,K12).EQ.KTRCH(K7,K13)) GOTO 24
      KNOT=KNOT+1
 24   CONTINUE
      IF (KNOT.NE.K8) GOTO 23
      KSUM=KSUM+1
      KSTORE(KSUM)=KTRCH(K10,K12)
 23   CONTINUE
      IF (KSUM.EQ.0) GOTO 22
      CONTINUE
      DO 25 K14=1,K8
      KSUM=KSUM+1
 25   KSTORE(KSUM)=KTRCH(K7,K14)
      CALL CHAINS(KSUM)
 22   CONTINUE
 21   CONTINUE
 12   CONTINUE
 11   CONTINUE
      WRITE(6,357)
      DO 310 N59=2,TR
      IF(KC(N59).EQ.0)GOTO 310
      N60=KC(N59)
      DO 311 N61=1,N60
      WRITE(6,309) (KCHAIN(N59,N61,I),I=1,N59)
 311  CONTINUE
 310  CONTINUE
      JTWO=0
      J=0
C COMBINE TRUNKCHAINS TO LARGER CHAINS
```

```
C   CALLED JTRCH
        DO 30 J1=2,TR
        IF (KC(J1).EQ.0) GOTO 30
        JKC=KC(J1)
        DO 31 J2=1,JKC
        IF (J.NE.0) GOTO 32
  33    JKL=1
        DO 34 J3=1,J1
  34    JTEMP1 (J3)=KCHAIN(J1,J2,J3)
        JT=J1
        GOTO 35
  32    CONTINUE
        IF (J.EQ.0) GOTO 610
        DO 608 J60=1,J
        J61=IABS(LENGTH(J60))
        LENGTH(J60)=J61
 608    CONTINUE
 610    CONTINUE
        DO 29 JCN=1,J1
  29    JTEMP1(JCN)=KCHAIN(J1,J2,JCN)
        JT=J1
        JD=0
        CALL NODUP
        JT=0
        JINT=0
        JTWO=0
        JFLAG1=0
        JFLAG2=0
        JMOR=0
        JNOTFL=0
        JF=0
        JCONT=0
        JL=0
C CHECK WHICH JTRCH'S CAN BE GROUPED TOGETHER
        DO 36 JC4=1,C
        J4=COMB(JC4)
        JLE=IABS(LENGTH(J4))
        LENGTH(J4)=JLE
        JF1=0
        JL1=0
        JCOUNT=0
        DO 37 J5=1,J1
  37    JTEMP(J5)=KCHAIN(J1,J2,J5)
        JN=0
        JFLAG=0
        JSTART=1
        DO 38 JV=1,J1
        IF(JTEMP(JV).EQ.JTRCH(J4,1)) JF1=JV
        CONTINUE
        IF(JTEMP(JV).EQ.JTRCH(J4,JLE)) JL1=JV
  38    CONTINUE
        IF (JF1.EQ.0. OR.JL1.EQ.0) GOTO 26
        LENGTH(J4)=-JLE
        JMOR=JMOR+1
  26    CONTINUE
        IF (JF1.NE.0) GOTO 39
        CONTINUE
        IF (JL1.NE.0) GOTO 40
        GOTO 41
  40    JL=JL+1
        JFLAG=1
        JLAST(JL)=J4
        JTEMP(JL1)=JTEMP(1)
        GOTO 42
  39    JF=JF+1
        JFLAG=2
        JFIRST(JF)=J4
        JTEMP(JF1)=JTEMP(1)
  42    CONTINUE
        JCOUNT=1
        JSTART=2
  41    CONTINUE
```

```
      JST=JSTART
      DO 43 J6=JST,JLE
      JR=J6
      IF(JFLAG.EQ.1) JR=JLE+1-J6
      CONTINUE
      DO 44 J7=JSTART,J1
      IF(JTEMP(J7).EQ.JTRCH(J4,JR))GOTO 45
 44   CONTINUE
      JN=JN+1
      IF(JN.EQ.(JLE-JST)) JNOTFL=JNOTFL+1
      GOTO 43
 45   JTEMP(J7)=JTEMP(JSTART)
      JSTART=JSTART+1
      IF(JCOUNT.EQ.0)GOTO 46
      JCOUNT=JCOUNT+1
      IF(JCOUNT.EQ.J6)GOTO 43
      JD=JD+1
      JDELET(JD)=J4
      GOTO 36
 46   JCOUNT=J6
 43   CONTINUE
      IF(JSTART.EQ.(J1+1)) GOTO 47
      CONTINUE
      IF (JFLAG.NE.0) GOTO 48
      IF (JNOTFL.GE.J) JFLAG2=1
      GOTO 36
 48   JINT=JINT+1
      JIC=0
      DO 49 J8 =JSTART,J1
      JIC=JIC+1
 49   JSTOR(JINT,JIC)=JTEMP(J8)
      JINTLE(JINT)=JIC
      JSEQ(JINT)=J4
      GOTO 36
 47   JCONT=1
      IF (JFLAG.EQ.2) GO TO 28
      CONTINUE
      IF(JFLAG.EQ.1) GO TO 27
      CONTINUE
      IF(JFLAG.EQ.0) GO TO 36
 27   JL=JL-1
      GOTO 36
 28   JF=JF-1
 36   CONTINUE
      IF(JD.EQ.0)GOTO 51
 50   CONTINUE
      JNEW=0
      DO 52 J9=1,J
      TEST=J9
      NEW=JNEW
      CALL JTRONE (NEW,TEST,&52)
      IF(JD.EQ.0)GOTO 52
      DO 53 J10=1,JD
      IF (J9.EQ.JDELET(J10))GOTO 52
 53   CONTINUE
      JNEW=JNEW+1
      J11=IABS(LENGTH(J9))
      DO 54 J12=1,J11
 54   JTRCH(JNEW,J12)=JTRCH(J9,J12)
      LENGTH(JNEW)=J11
 52   CONTINUE
      J=JNEW
      IF(JFLAG1.EQ.1)GOTO 31
 51   CONTINUE
C KCHAIN IS COMPLETELY CONTAINED IN A JTRCH
      IF (JCONT.EQ.1) GOTO 31
      CONTINUE
      IF (JFLAG2.EQ.1) GOTO 33
      CONTINUE
C THREE OR MORE JTRCH ARE COMBINED
      IF((JF+JL+JMOR).LE.2)GO TO 55
      KTOTLE=0
      DO 56   K30=1,JINT
      K31=JSEQ(K30)
      K32=IABS(LENGTH(K31))
56    KTOTLE=KTOTLE+K32
      JNTR=0
      IF (KTOTLE.EQ.J1) GOTO 57
      J=J+1
      DO 58   K33=1,J1
      KNOT=0
      DO 59   K34=1,JINT
      K35=JSEQ(K34)
      K36=IABS(LENGTH(K35))
      DO 60   K37=1,K36
      IF (KCHAIN(J1,J2,K33).EQ.JTRCH(K35,K37)) GOTO 60
      KNOT=KNOT+1
60    CONTINUE
59    CONTINUE
      IF (KNOT.NE.KTOTLE) GOTO 58
      JNTR=JNTR+1
      JTRCH(J,JNTR)=KCHAIN(J1,J2,K33)
58    CONTINUE
      JINT=JINT+1
      JSEQ(JINT)=J
      LENGTH(J)=JNTR
57    CONTINUE
      DO 61   K38=1,JINT
      K39=JSEQ(K38)
      K40=IABS(LENGTH(K39))
      JD=JD+1
      JDELET(JD)=K39
      J=J+1
      LONG=0
      DO 62   K41=1,K40
      LONG=LONG+1
62    JTRCH(J,LONG)=JTRCH(K39,K41)
      K42=JINT-1
      DO 63   K43=1,K42
      KL=K38+K43
      IF (KL.GT.JINT) KL=KL-JINT
      K44=JSEQ(KL)
      K45=IABS(LENGTH(K44))
      DO 63   K46=1,K45
      LONG=LONG+1
63    JTRCH(J,LONG)=JTRCH(K44,K46)
      LENGTH(J)=LONG
61    CONTINUE
      IF (JNTR.LE.2) GOTO 64
      JPREV=J
      JPLE=LENGTH(J)
      JMOREP=(JNTR-2)/2
      JODD=0
      IF(((JNTR-1)/2).EQ.(JNTR/2)) JODD=JNTR-JNTR/2
      CONTINUE
      IF (JODD.NE.0) JMOREP=JMOREP+1
      CONTINUE
      DO 65   K47=1,JMOREP
      J=J+1
      K49=JSEQ(JINT)
      K50=LENGTH(K49)
      K51=JTRCH(K49,1+K47)
      K52=JTRCH(K49,K50-K47)
      IF (K47.EQ.JMOREP.AND.JODD.NE.0) K51=JTRCH(K49,JODD)
      CONTINUE
      IF (K47.EQ.JMOREP.AND.JODD.NE.0) K52=JTRCH(K49,1)
      JTRCH(J,1)=K51
      COUNT=1
      DO 66   K53=1,K50
      IF (JTRCH(K49,K53).EQ.K51.OR.JTRCH(K49,K53).EQ.K52)
     1 GOTO 66
      COUNT=COUNT+1
      JTRCH(J,COUNT)=JTRCH(K49,K53)
66    CONTINUE
```

```
      K54=K50+1
      DO 67  K55=K54,JPLE
      COUNT=COUNT+1
 67   JTRCH(J,COUNT)=JTRCH(JPREV,K55)
      JTRCH(J,JPLE)=K52
 65   LENGTH(J)=JPLE
 64   JFLAG1=1
      GOTO 50
C TWO JTRCH ARE COMBINED,JKL INDICATES HOW
 55   J22=JINTLE(1)
      IF (JINT.GT.1) GOTO 68
      JT=J22
      DO 69 J23=1,J22
 69   JTEMP1(J23)=JSTOP(1,J23)
      GOTO 70
 68   J24=JINTLE(2)
      JT =0
      DO 71 J25=1,J22
      DO 72 J26=1,J24
      IF (JSTOR(1,J25).NE.JSTOR(2,J26)) GOTO 64
      JT=JT+1
      JTEMP1(JT)=JSTOR(1,J25)
 72   CONTINUE
 71   CONTINUE
 70   CONTINUE
      IF(JNOTFL.GE.J)JKL=1
      CONTINUE
      IF(JF.EQ.0.AND.JL.EQ.1)JKL=2
      CONTINUE
      IF(JF.EQ.1.AND.JL.EQ.0)JKL=3
      CONTINUE
      IF(JF.EQ.1.AND.JL.EQ.1)JKL=4
      CONTINUE
      IF(JF.EQ.0.AND.JL.EQ.2)JKL=5
      CONTINUE
      IF(JF.EQ.2.AND.JL.EQ.0)JKL=6
      CONTINUE
      IF (JKL.GE.4) JTWO=JTWO+1
 35   CONTINUE
      CALL PERMUT(JKL,JNLAST,&74)
      CONTINUE
C A NEW JTRCH IS GENERATED FROM A CHAIN
      DO 73 J27=1,JNLAST
      J=J+1
      LENGTH(J)=JT
      DO 73 J28=1,JT
 73   JTRCH(J,J28)=JADD(J27,J28)
      GOTO 31
 74   MULT=JNLAST
      IF (JNLAST.EQ.0) MULT=1
      FWD1=MULT
      J29=JLAST(1)
      JD=JD+1
      JDELET(JD)=J29
      IF (JTWO.EQ.0) GOTO 75
      J30=JFIRST(1)
      JD=JD+1
      JDELET(JD)=J30
      IF (LENGTH(J30).LT.0) FWD1=MULT*2
      J32=IABS(LENGTH(J30))
 75   TIMES=1
      IF (LENGTH(J29).LT.0) TIMES=2
      J31=IABS(LENGTH(J29))
      JAD=0
      JFC=0
      DO 76 J33=1,TIMES
      DO 77 J34=1,FWD1
      J=J+1
      DO 78 J35=1,J31
      JLD=J35
      IF (J33.EQ.2) JLD=J31+1-J35
 78   JTRCH(J,J35)=JTRCH(J29,JLD)
```

```
      IF (JNLAST.EQ.0) GOTO 79
      JAD=JAD+1
      IF (JAD.GT.JNLAST) JAD=JAD-JNLAST
      CONTINUE
      DO 80 J36=1,JT
 80   JTRCH(J,J31+J36)=JADD(JAD,J36)
 79   CONTINUE
      IF (JTWO.EQ.0) GOTO 81
      JFC=JFC+1
      IF (JFC.GT.FWD2) JFC=JFC-FWD2+1
      DO 82 J37=1,J32
      JFD=J37
      IF (JFC.LE.0) JFD=J32+1-J37
 82   JTRCH(J,J31+JT+J37)=JTRCH(J30,JFD)
 81   CONTINUE
      LENGTH(J)=J31+JT
      IF (JTWO.NE.0) LENGTH(J)=LENGTH(J) + J32
 77   CONTINUE
 76   CONTINUE
      JFLAG1=1
      GOTO 50
 31   CONTINUE
 30   CONTINUE
      T=0
      DO 83 J42=1,J
      IF (LENGTH(J42).NE.TR) GOTO 83
      T=T+1
      J43=LENGTH(J42)
      DO 84 J44=1,J43
 84   TRUNK(T,J44)=JTRCH(J42,J44)
 83   CONTINUE
      WRITE(6,358) T
      DO 312 N62=1,T
      WRITE(6,309) (TRUNK(N62,I),I=1,TR)
 312  CONTINUE
C FOR EACH MAIN BRANCH THE LOOPS IT APPEARS IN
C     ARE LISTED,ALSO THE OTHER TRUNK AND
C   MAIN BRANCHES PRESENT IN THESE LOOPS
      DO  85 M1=1,MBR
      M2=MAINBR(M1)
      MS=0
      DO  86 M3=2,ROW
      IF (F(M3,M2).EQ.0) GOTO 86
      MS=MS+1
      MST=0
      DO  87 M4=1,TR
      M5=TRUNBR(M4)
      IF (F(M3,M5).EQ.0) GOTO 87
      MST=MST+1
      MBRT(M1,MS,MST)=F(1,M5)
 87   CONTINUE
      MSLET(M1,MS)=MST
      MSB=0
      DO  88 M6=1,MBR
      M7=MAINBR(M6)
      IF (F(M3,M7).EQ.0.OR.M6.EQ.M1) GOTO 88
      MSB=MSB+1
      MBRB(M1,MS,MSB)=F(1,M7)
 88   CONTINUE
      MSLEB(M1,MS)=MSB
 86   CONTINUE
      MSER(M1)=MS
 85   CONTINUE
      DO  89 M8=1,T
      IF (M8.GT.NOGRAF) GOTO 89
      CONTINUE
      DO 217 MREP=1,MBR
 217  MSER(MREP)=IABS(MSER(MREP))
      ME=0
      MU=0
      MA=0
C FIND UNIQUE CONNECTIONS TO A NODE BETWEEN
```

```
C  TWO TRUNK BRANCHES
      DO  90 M10=1,MBR
      MCOL=MAINBR(M10)
      M11= MSER(M10)
      MFIRST=0
      TR1=TR-1
      DO  91 M14=1,TR1
      M15=TRUNK(M8,M14)
      NEX=TRUNK(M8,M14+1)
      MFS=0
      MSS=0
      ISUNIQ=0
      LWTBR=0
      DO  92 M16=1,M11
      IF (MSLET(M10,M16).EQ.0) GOTO 92
      LWTBR=LWTBR+1
      MFIRST=0
      MSECON=0
      M17=MSLET(M10,M16)
      DO  93 M18=1,M17
      IF (M15.EQ.MBRT(M10,M16,M18)) MFIRST=1
      CONTINUE
      IF (M15.EQ.MBRT(M10,M16,M18)) MFS=MFS+1
      CONTINUE
      IF (NEX.EQ.MBRT(M10,M16,M18)) MSECON=1
      CONTINUE
      IF (NEX.EQ.MBRT(M10,M16,M18)) MSS=MSS+1
 93   CONTINUE
      IF ((MFIRST+MSECON).NE.1)  GOTO 91
      ISUNIQ=ISUNIQ+1
 92   CONTINUE
      IF (MFS.EQ.0.OR.MSS.EQ.0) GOTO 91
      CONTINUE
      IF (ISUNIQ.EQ.LWTBR.AND.LWTBR.NE.0) GOTO 94
 91   CONTINUE
      GOTO 90
 94   CONTINUE
      MT(1)=0
      MT(2)=0
      DO 106 M19=1,M11
      IF (MSLET(M10,M19).EQ.0) GOTO 106
      M20= MSLET(M10,M19)
      DO  95 M21=1,M20
      IF (MBRT(M10,M19,M21).EQ.M15) GOTO 96
      CONTINUE
      IF (MBRT(M10,M19,M21).NE.NEX) GOTO 95
      MT(2)=MT(2)+1
      IF (MSLEB(M10,M19).EQ.0) GOTO 105
      MY=MT(2)
      MTS(2,MY)=M19
      GOTO 106
 96   MT(1)=MT(1)+1
      IF (MSLEB(M10,M19).EQ.0) GOTO 105
      MX=MT(1)
      MTS(1,MX)=M19
 95   CONTINUE
 106  CONTINUE
      MBG=0
      DO  97 M22=1,2
      NBG=0
      M23=MTS(M22,1)
      M24=MSLEB(M10,M23)
      DO  98 M25=1,M24
      MCOT=0
      IF(MT(M22).EQ.1) GOTO 100
      M26=MT(M22)
      DO  99 M27=1,M26
      M28=MTS(M22,M27)
      M29=MSLEB(M10,M28)
      DO  99 M30=1,M29
      IF(MBRB(M10,M23,M25).EQ.MBRB(M10,M28,M29))MCOT=MCOT+1
 99   CONTINUE
```

```
      IF(MCOT.EQ.(M26-1))GO TO 100
      GOTO 98
  100 NBG=NBG+1
      MCOM(M22,NBG)=MBRB(M10,M23,M25)
      MC(M22)=NBG
   98 CONTINUE
   97 CONTINUE
      IF(MC(1).EQ.0.OR.MC(2).EQ.0)GO TO 105
      M31=MC(1)
      M32=MC(2)
      DO 101 M33=1,M31
      DO 102 M34=1,M32
      IF(MCOM(1,M33).EQ.MCOM(2,M34)) GO TO 103
  102 CONTINUE
      GO TO 101
  103 MBG=MBG+1
      MBGS(MBG)=MCOM(1,M33)
  101 CONTINUE
      IF (MBG.EQ.0) GOTO 105
      MA=MA+1
      MAMBIG(MA,1)=F(1,MCOL)
      MAMBIG(MA,2)=MBG
      MAMBIG(MA,3)=NEX
      MAMBIG(MA,4)=M10
      MAMBIG(MA,5)=1
      DO 104 M35=1,MBG
  104 MAMBIG(MA,5+M35)=MBGS(M35)
      KA=MBG+5
      GOTO 90
  105 MU=MU+1
      MUNIQ(MU,1)=F(1,MCOL)
      MUNIQ(MU,2)=NEX
      MUNIQ(MU,3)=M10
      MUNIQ(MU,4)=1
   90 CONTINUE
C FIND UNIQUE CONNECTION TO ONE OF THE
C  ENDNODES OF A PATH OF TRUNK BRANCHES
      NSIMP=0
      NSEP=0
      DO 107 M39=1,TR
      MCP=0
      M40=TRUNK(M8,M39)
      DO 108 M41=1,MBR
      CHECK=0
      RELPOS=1
      MEISER=0
      MCOL=MAINBR(M41)
       MUFLA=0
      MCOT=0
      MCOT1=0
      M42=MSER(M41)
      DO 109 M43=1,M42
      IF(MSLET (M41,M43).EQ.0)GO TO 109
      M44=MSLET(M41,M43)
      IF (M44.GT.1)      MEISER=1
      MCOT1=MCOT1+1
      DO 110 M45=1,M44
      IF (CHECK.EQ.1) GOTO 229
      CONTINUE
      DO 228 N75=1,TR
      N76=TRUNK(M8,N75)
      IF (N76.EQ.M40) GOTO 228
      CONTINUE
      IF(MBRT(M41,M43,M45).NE.N76) GOTO 228
      CHECK=1
      IF (N75.LT.M39) RELPOS=2
      GOTO 229
  228 CONTINUE
  229 CONTINUE
      IF(MBRT(M41,M43,M45).NE.M40)GO TO 110
      MCOT=MCOT+1
      IF(MSLEB(M41,M43).EQ.0) MUFLA=1
```

```
110   CONTINUE
109   CONTINUE
      IF (MEISER.EQ.0) GOTO 108
      CONTINUE
      IF(MCOT1.EQ.MCOT.AND.MCOT.NE.0)GO TO 111
      GO TO 108
111   MCP=MCP+1
      IF (MUFLA.EQ.1) GOTO 112
      MCOMP(MCP)=M41
      GO TO 108
112   CONTINUE
      MU=MU+1
      MCP=0
      MUNIQ(MU,1)=F(1,MCOL)
      MUNIQ(MU,2)=M40
      MUNIQ(MU,3)=M41
      MUNIQ(MU,4)=RELPOS
108   CONTINUE
      IF (MCP.EQ.0) GOTO 107
      CONTINUE
      DO 113 M47=1,MCP
      MC1=0
      MBG=0
      M48=MCOMP(M47)
      M49=MSER(M48)
      IF(M49.NE.1)GO TO 114
      MC1=MSLEB(M48,1)
      DO 115 M36=1,MC1
115   MCOM(1,M36)=MBRB(M48,1,M36)
      GO TO 122
114   CONTINUE
      DO 116 M50=1,M49
      IF (MSLET(M48,M50).EQ.0) GOTO 116
      M51=MSLET(M48,M50)
      DO 117 M52=1,M51
      IF(MBRT(M48,M49,M52).EQ.M40)GO TO 118
117   CONTINUE
      GO TO 116
118   MBG=MBG+1
      MBGS(MBG)=M50
116   CONTINUE
      M54=MBGS(1)
      M55=MSLEB(M48,M54)
      DO 119 M56=1,M55
      MCOT=0
      M57=MBRB(M48,M54,M56)
      DO 120 M58=2,MBG
      M59=MBGS(M58)
      M60=MSLEB(M48,M59)
      DO 121 M61=1,M60
      IF(MBRB(M48,M59,M61).NE.M57) GO TO 121
      MCOT=MCOT+1
121   CONTINUE
120   CONTINUE
      IF(MCOT.NE.(MBG-1))GO TO 119
      MC1=MC1+1
      MCOM(1,MC1)=M57
119   CONTINUE
      IF(MC1.NE.0)GO TO 122
      MU=MU+1
      MUNIQ(MU,1)=F(1,MCOL)
      MUNIQ(MU,2)=M40
      MUNIQ(MU,3)=M41
      MUNIQ(MU,4)=3
      IF (M39.NE.1.AND.M39.NE.TR) MUNIQ(MU,4)=RELPOS
      CONTINUE
      GOTO 113
122   MA=MA+1
      MCOL=MAINBR(M48)
      MAMBIG(MA,1)= F(1,MCOL)
      MAMBIG(MA,2)=MC1
      MAMBIG(MA,3)=M40
      MAMBIG(MA,4)=M48
      MAMBIG(MA,5)=3
      IF (M39.NE.1.AND.M39.NE.TR) MAMBIG(MA,5)=RELPOS
      CONTINUE
      DO 123 M62=1,MC1
  123 MAMBIG(MA,5+M62)=MCOM(1,M62)
      KA=MC1+5
  113 CONTINUE
  107 CONTINUE
      MGR=0
  124 CONTINUE
C STORE CONNECTIONS IN GRAPH
      IF (MU.EQ.0) GOTO133
      CONTINUE
      IF (MA.EQ.0) GOTO126
      MCT=0
      DO 125 M65=1,MU
      IF(MCT.EQ.MA)GOTO 126
      CONTINUE
      DO 127 M66=1,MA
      IF(MAMBIG(M66,1).EQ.0)GO TO 127
      M67=MAMBIG(M66,2)+5
      DO 128 M68=6,M67
      IF (MAMBIG(M66,M68).EQ.MUNIQ(M65,1)) GOTO 130
  128 CONTINUE
      GO TO 127
  130 MAMBIG(M66,1)=0
      MCT=MCT+1
  127 CONTINUE
  125 CONTINUE
  126 CONTINUE
      DO 131 M69=1,MU
      MGR=MGR+1
      GRAPH(MGR,1)=MUNIQ(M69,1)
      GRAPH(MGR,2)=MUNIQ(M69,2)
      GRAPH(MGR,3)=MUNIQ(M69,4)
      M70=MUNIQ(M69,3)
      ME=ME+1
      MEND(ME)=MUNIQ(M69,1)
  131 MSER(M70)=-MSER(M70)
  133 MU=0
      IF (MA.EQ.0) GOTO 138
      CONTINUE
      DO 134 M71=1,MA
      IF(MAMBIG(M71,1).EQ.0)GO TO 134
      CONTINUE
      DO 141 MAC=1,MGR
      IF (GRAPH(MAC,1).EQ.MAMBIG(M71,1)) GOTO 134
  141 CONTINUE
      MGR=MGR+1
      GRAPH(MGR,1)=MAMBIG(M71,1)
      GRAPH(MGR,2)=MAMBIG(M71,3)
      GRAPH(MGR,3)=MAMBIG(M71,5)
      M72=MAMBIG(M71,4)
      ME=ME+1
      MEND(ME)=MAMBIG(M71,1)
      MSER(M72)=-MSER(M72)
      M73=MAMBIG(M71,2)+5
      DO 136 M74=6,M73
      DO 129 MAK=1,MGR
      IF (GRAPH(MAK,1).EQ.MAMBIG(M71,M74)) GOTO 136
  129 CONTINUE
      MGR=MGR+1
      GRAPH(MGR,1)=MAMBIG(M71,M74)
      GRAPH(MGR,2)=GRAPH(MGR-1,1)
      GRAPH (MGR,3)=7
      DO 137 M75=1,MBR
      M76=MAINBR(M75)
      IF (F(1,M76).NE.MAMBIG(M71,M74))GOTO 137
      MSER(M75)=-MSER(M75)
      ME=ME+1
      MEND(ME)=MAMBIG(M71,M74)
```

```
 137  CONTINUE
 136  CONTINUE
 134  CONTINUE
 138  MA=0
 139  NSIMP=0
 140  CONTINUE
C FIND CONNECTIONS TO ALREADY CONNECTED
C  MAINBRANCHES.IF CONNECTION IS UNIQUE
C  NSIMP IS ZERO
      CCT=0
      DO 142 N1=1,ME
      N3=MEND(ME+1-N1)
      MCP=0
      DO 143 N4=1,MBR
      IF (MSER(N4).LE.0) GOTO 143
      MCOT1=0
      MCOT=0
      N5=MSER(N4)
      DO 144 N6=1,N5
      IF (MSLEB(N4,N6).EQ.0) GOTO 144
      MCOT1=MCOT1+1
      N7=MSLEB(N4,N6)
      DO 145 N8=1,N7
      IF (MBRB(N4,N6,N8).NE.N3) GOTO145
      MCOT=MCOT+1
      IF (NSIMP.EQ.10) GOTO 146
      CONTINUE
      IF (N7.EQ.1) GOTO 147
 145  CONTINUE
 144  CONTINUE
      IF (MCOT1.EQ.MCOT.AND.MCOT.NE.0) GOTO 146
      CONTINUE
      IF (MCOT.NE.0) GOTO 146
      GOTO 143
 146  MCP=MCP+1
      MCOMP(MCP)=N4
      GOTO 143
 147  CONTINUE
      IF (CCT.EQ.0) GOTO 210
      DO 209 C1=1,CCT
      IF (N4.EQ.CONECT(C1)) GOTO 211
 209  CONTINUE
 210  CONTINUE
      CCT= CCT+1
      CONECT(CCT)=N4
      MU=MU+1
      N8=MAINBR(N4)
      MUNIQ(MU,1)=F(1,N8)
      MUNIQ(MU,2)=N3
      MUNIQ(MU,3)=N4
      MUNIQ(MU,4)=3+NSIMP
 211  CONTINUE
 143  CONTINUE
      IF(MCP.EQ.0) GOTO 142
      CONTINUE
      DO 148 N9=1,MCP
      MC1=0
      MBG=0
      N10=MCOMP(N9)
      MCOL=MAINBR(N10)
      N11=MSER(N10)
      IF (N11.NE.1) GOTO 150
      MC1=MSLEB(N10,1)
      NIC=0
      DO 149 N12=1,MC1
      IF(MBRB(N10,1,N12).EQ.N3) GOTO 149
      NIC=NIC+1
      MCOM(1,NIC)=MBRB(N10,1,N12)
 149  CONTINUE
      MC1=NIC
      GOTO 159
 150  CONTINUE
```

```
      DO 151 N13=1,N11
      IF (MSLEB(N10,N13).EQ.0) GOTO 151
      N14=MSLEB(N10,N13)
      DO 152 N15=1,N14
      IF (MBRB(N10,N13,N15).EQ.N3) GOTO 153
  152 CONTINUE
      GOTO 151
  153 MBG=MBG+1
      MBGS(MBG)=N13
  151 CONTINUE
      IF (MBG.EQ.1) GOTO 161
      N16=MBGS(1)
       N17=MSLEB(N10,N16)
      DO 154 N18=1,N17
      MCOT=0
      N19=MBRB(N10,N16,N18)
      IF (N19.EQ.N3) GOTO 154
      CONTINUE
      DO 155 N20=1,MBG
      N21=MBGS(N20)
      N22=MSLEB(N10,N21)
      DO 156 N23=1,N22
      IF(MBRB(N19,N21,N23).NE.N19) GOTO 156
      MCOT=MCOT+1
  156 CONTINUE
  155 CONTINUE
      IF(MCOT.NE.(MBG-1)) GOTO 154
      MC1=MC1+1
      MCOM(1,MC1)=N19
  154 CONTINUE
      IF(MC1.EQ.0) GOTO 161
  159 CONTINUE
      IF (CCT.EQ.0) GOTO 212
      DO 213 C2=1,CCT
      IF(N10.EQ.CONECT(C2)) GOTO 148
  213 CONTINUE
  212 CONTINUE
      CCT= CCT+1
      CONECT(CCT)=N10
      MA=MA+1
      MAMBIG(MA,1)=F(1,MCOL)
      MAMBIG(MA,2)=MC1
      MAMBIG(MA,3)=N3
      MAMBIG(MA,4)=N10
      MAMBIG(MA,5)=3+NSIMP
      DO 160 N24=1,MC1
  160 MAMBIG(MA,5+N24)=MCOM(1,N24)
      KA=MC1+5
      GOTO 148
  161 CONTINUE
      MU=MU+1
      IF (CCT.EQ.0) GOTO 214
      CONTINUE
      DO 215 C3=1,CCT
      IF(N10.EQ.CONECT(C3)) GOTO 148
  215 CONTINUE
  214 CONTINUE
      MUNIQ(MU,1)=F(1,MCOL)
      MUNIQ(MU,2)=N3
      MUNIQ(MU,3)=N10
      MUNIQ(MU,4)=3+NSIMP
  148 CONTINUE
  142 CONTINUE
      IF((MU+MA).NE.0) GOTO 124
      MCOUNT=0
      DO 163 N25=1,MBR
      IF(MSER(N25).LE.0) GOTO 173
      MCOL=MAINBR(N25)
      IF(NSIMP.NE.0) GOTO 164
      NSIMP=10
C NO UNIQUE CONNECTION COULD BE MADE
      GOTO 140
```

```
C CONNECT SO AS TO FORM A LINEAR PATH WITH
C   MAIN BRANCHES
 164   N29=MSER(N25)
       DO 166 N26=1,MGR
       N27=MGR+1-N26
       N28=GRAPH(N27,1)
       DO 168 N30=1,N29
       IF(MSLEB(N25,N30).EQ.0)GOTO 168
       N31=MSLEB(N25,N30)
       DO 169 N32=1,N31
       IF(MBRB(N25,N30,N31).NE.N28) GOTO 169
       MU=MU+1
       MUNIQ(MU,1)=F(1,MCOL)
       MUNIQ(MU,2)=N28
       MUNIQ(MU,3)=N25
       MUNIQ(MU,4)=3+NSIMP
       GOTO 126
 169   CONTINUE
 168   CONTINUE
 166   CONTINUE
C CONNECT SO AS TO FORM A LINEAR PATH WITH
C   TRUNK BRANCHES
       DO 170 N34=1,TR
       N38=TRUNK(M8,N34)
       DO 171 N35=1,N29
       IF(MSLET(N25,N35).EQ.0)GOTO 171
       N36=MSLET(N25,N35)
       DO 172 N37=1,N36
       IF(MBRT(N25,N35,N37).NE.N38) GOTO 172
       MU=MU+1
       MUNIQ (MU,1)=F(1,MCOL)
       MUNIQ (MU,2)=N38
       MUNIQ (MU,3)=N25
       MUNIQ(MU,4)=1+NSIMP
       GOTO 126
 172   CONTINUE
 171   CONTINUE
 170   CONTINUE
C MAIN BRANCH STARTS A SEPARATE PART
       MU=MU+1
       MUNIQ(MU,1)=F(1,MCOL)
       MUNIQ(MU,2)=0
       MUNIQ(MU,3)=N25
       MUNIQ(MU,4)=6
       GOTO 126
 173   MCOUNT=MCOUNT+1
       IF(MCOUNT.EQ.MBR) GOTO 216
 163   CONTINUE
 216   CONTINUE
       IF (L.EQ.0) GOTO 175
       CONTINUE
       CALL LIMBS
 175   CONTINUE
       CALL LINKS
       CONTINUE
       CALL OUTPUT
 89    CONTINUE
 350   FORMAT('1',T30,'F-MATRIX'/)
 351   FORMAT(//T30,'F-MATRIX,PARTITIONED'/)
 353   FORMAT(//1H ,T30,'MAINLOOP IS ROW # ',I2)
 354   FORMAT (//26X,I2,T30,'TRUNKBRANCHES')
 359   FORMAT(1H ,27X,20I3)
 355   FORMAT  (/26X,I2,T30,'MAINBRANCHES')
 356   FORMAT  (/26X,I2,T30,'LIMBS')
 357   FORMAT(//T30,'COMMON NODE REQUIREMENTS')
 358   FORMAT  (//26X,I2,T30,'TRUNKS')
       STOP
       END


       SUBROUTINE OUTPUT
C THIS SUBROUTINE CONVERTS NUMBERCODE FOR
```

```
C  CONNECTIONS INTO WORDS
        IMPLICIT INTEGER (A-Z)
      DIMENSION F(30,30),LIMB(20),NOTLIM(20),MAINBR(20)
      DIMENSION TRUNBR(20)
      DIMENSION MEND(30),GRAPH(60,3),TRUNK(20,20),FPART(30)
      COMMON/LLO/ ROW,COL,F,TR,TRUNBR,M8,TRUNK,MBR,MAINBR,
     1MEND,MGR,GRAPH,L,LIMB,ME
      WRITE(6,359) M8
      WRITE(6,309) (TRUNK(M8,I),I=1,TR)
      WRITE(6,360)
      IF(MBR.NE.0) WRITE(6,343)
      MOTHER=10
      DO 314 N63=1,MGR
      MOTHER=MOTHER+1
      IF(N63.EQ.(MBR+1)) WRITE(6,344)
      CONTINUE
      IF(N63.EQ.(MBR+L+1))WRITE(6,345)
      CONTINUE
      IF(N63.EQ.(MBR+L+1))MOTHER=-1
      CONTINUE
      IF(MOTHER.EQ.0) GO TO 341
      CONTINUE
      IF(MOTHER.GT.10) GOTO 361
      MOTHER=-1
 361  WRITE(6,315)GRAPH(N63,1)
      GOTO 346
 341  WRITE(6,342)
 346  LOKT=GRAPH(N63,3)
      LOCT=LOKT
      IF (LOKT.GT.7) LOCT=LOCT-10
      CONTINUE
      IF (LOCT.EQ.1) GOTO 321
      CONTINUE
      IF (LOCT.EQ.2) GOTO 322
      CONTINUE
      IF (LOCT.EQ.3) GOTO 323
      CONTINUE
      IF (LOCT.EQ.4) GOTO 324
      CONTINUE
      IF (LOCT.EQ.5) GOTO 325
      CONTINUE
      IF (LOCT.EQ.6) GOTO 326
      CONTINUE
      IF (LOCT.EQ.7) GOTO 327
 321  WRITE(6,331)
      GOTO 316
 322  WRITE(6,332)
      GOTO 316
 323  WRITE(6,333)
      GO TO 316
 324  WRITE(6,334)
      GO TO 316
 325  WRITE(6,335)
      GOTO 316
 326  WRITE(6,336)
      GOTO 314
 327  WRITE(6,337)
      WRITE(6,318) GRAPH(N63-1,1)
      WRITE(6,339)
      GOTO 314
 316  CONTINUE
      WRITE(6,317) GRAPH(N63,2)
      IF (LOKT.GT.7) WRITE (6,338)
 314  CONTINUE
 315  FORMAT   (/31X,I2)
 317  FORMAT(1H+,52X,I2)
 318  FORMAT(1H+,49X,I2)
 331  FORMAT('+',T39,'LEFT OF')
 332  FORMAT('+',T39,'RIGHT OF')
 333  FORMAT('+',T39,'OUTSIDE OF')
 334  FORMAT('+',T39,'INSIDE OF')
 335  FORMAT('+',T39,'EITHER SIDE OF')
 309   FORMAT(1H ,27X,20I3)
 336   FORMAT('+',T39,'STARTS A SEPARATE PART')
 337   FORMAT('+',T39,'AND BRANCH#')
 339   FORMAT('+',T54,'INTERCHANGEABLE')
 338   FORMAT('+',T61,'(NOT UNIQUE)')
 342   FORMAT(T3,' ')
 343   FORMAT(/T30,'MAIN:')
 300   FORMAT (1H ,28X,30I2)
 344   FORMAT(/T30,'LIMB:')
 345   FORMAT(//T30,'LINK:',T39,'FROM/TO',T53,'BRANCH'/)
 359   FORMAT(//T30,'GRAPH FOR TRUNK',I2,1H:)
 360   FORMAT(//T30,'BRANCH',T39,'SIDE OF',T53,'BRANCH')
       RETURN
       END


       SUBROUTINE LINKS
C THE POSITION RELATIVE TO THE TREE IS FOUND.
C  THE CONNECTION PATTERM IS INDICATED BY CASE
         IMPLICIT INTEGER (A-Z)
       DIMENSION F(30,30),LIMB(20),NOTLIM(20),MAINBR(20)
       DIMENSION TRUNBR(20)
       DIMENSION MEND(30),GRAPH(60,3),TRUNK(20,20),FPART(30)
       COMMON/LLO/ ROW,COL,F,TR,TRUNBR,M8,TRUNK,MBR,MAINBR,
      1MEND,MGR,GRAPH,L,LIMB,ME
       TGR=0
       DO 183 N50=2,ROW
       TSUM=0
       IF (TR.EQ.0) GOTO 184
       CONTINUE
       DO 185 N51=1,TR
       N52=TRUNBR(N51)
       IF (F(N50,N52).EQ.1) TSUM=TSUM+1
 185   CONTINUE
 184   MSUM=0
       IF (MBR.EQ.0) GOTO 186
        DO 187 N53=1,MBR
       N54=MAINBR(N53)
       IF (F(N50,N 54).EQ.1) MSUM=MSUM+1
 187   CONTINUE
 186   LSUM=0
       IF (L.EQ.0) GOTO 188
       DO 189 N55=1,L
       N56=LIMB(N55)
       IF (F(N50,N56).EQ.1) LSUM=LSUM+1
 189   CONTINUE
 188   CASE=6
       IF (LSUM.NE.0) CASE=CASE-3
       CONTINUE
       IF (MSUM.EQ.0.AND.TSUM.EQ.0) CASE=CASE-3
       CONTINUE
       IF (MSUM.NE.0) CASE=CASE-1
       CONTINUE
       IF (TSUM.NE.0.AND.MSUM.NE.0) CASE=CASE-1
       CONTINUE
       IF (CASE.EQ.6) GOTO 196
       CONTINUE
       IF (CASE.NE.4) GOTO 230
       NT=0
       DO 227 NS1=1,MBR
       NS2=MAINBR(NS1)
       IF (F(N50, NS2).EQ.0) GOTO 227
       CONTINUE
       DO 224 NS3=1,MGR
       IF (GRAPH(NS3,2).EQ.F(1,NS2)) GOTO 227
 224   CONTINUE
       NT=NT+1
       IF (NT.EQ.2) GOTO 225
       NTMB=F(1,NS2)
       GOTO 227
 225   MGR=MGR+1
       GRAPH(MGR,1)=F(N50,1)
      GRAPH(MGR,2)=NTMB
      GRAPH(MGR,3)=3
      MGR=MGR+1
      GRAPH(MGR,1)=F(N50,1)
      GRAPH(MGR,2)=F(1,N52)
      GRAPH(MGR,3)=3
      GOTO 183
227   CONTINUE
230   CONTINUE
      IF (CASE.GE.4) GOTO 190
      CONTINUE
      DO 191 N57=1,L
      N58=LIMB(L+1-N57)
      IF (F(N50,N58).EQ.0) GOTO 191
      DO 192 N70=1,MGR
      IF (GRAPH(MGR+1-N70,1).NE.F(1,N58)) GOTO 192
      MG=MGR+1
      GRAPH(MG ,1)=F(N50,1)
      GRAPH(MG ,2)=F(1,N58)
      GRAPH(MG ,3)=3
      GOTO 226
192   CONTINUE
191   CONTINUE
226   MGR=MGR+1
      IF (CASE.EQ.1) GOTO 194
      CONTINUE
      IF (CASE.EQ.2) GOTO 190
      CONTINUE
      DO 193 N71=1,L
      N72=LIMB(N71)
      IF (F(N50,N72).EQ.0) GOTO 193
      CONTINUE
      DO 195 N69=1,MGR
      IF (GRAPH(MGR+1-N69,1).NE.F(1,N72)) GOTO 195
      CONTINUE
      IF (GRAPH(MGR+1-N69,3).EQ.7.AND.CASE.EQ.3) GOTO 195
      CONTINUE
      IF (CASE.EQ.3)GOTO 197
      MG=MGR+1
      GRAPH(MG ,1)=F(N50,1)
      GRAPH(MG ,2)=F(1,N72)
      GRAPH(MG ,3)=3
      GOTO 197
195   CONTINUE
193   CONTINUE
197   TGR=GRAPH(MGR+1-N69,2)
      IF (CASE.EQ.0) MGR=MGR+1
      CONTINUE
      IF (CASE.EQ.0) GOTO 183
      IF (GRAPH(MGR+1-N69,3).EQ.1) POSONE=2
      CONTINUE
      IF (GRAPH(MGR+1-N69,3).EQ.2) POSONE=1
      GOTO 196
190   MLOW=100
      MHIGH=0
      DO 198 N59=1,MBR
      N60=MAINBR(N59)
      IF (F(N50,N60).EQ.0) GOTO 198
      CONTINUE
      DO 199 N61=1,ME
      IF (MEND(N61).NE.F(1,N60)) GOTO 199
      CONTINUE
      IF (N61.LT.MLOW) MLOW=N61
      CONTINUE
      IF (N61.GT.MHIGH) MHIGH=N61
199   CONTINUE
198   CONTINUE
      IF (CASE-4) 200,201,200
200   MGR=MGR+1
      GRAPH(MGR,1)=F(N50,1)
      GRAPH(MGR,3)=4
      GRAPH(MGR,2)=MEND(MLOW)
      IF (CASE.EQ.2) GOTO 183
  201 MGR=MGR+1
      GRAPH(MGR,1)=F(N50,1)
      GRAPH(MGR,3)=3
      GRAPH(MGR,2)=MEND(MHIGH)
      IF (CASE.EQ.5) GOTO 183
  194 CONTINUE
      DO 202 N62=1,MBR
      N63=MAINBR(N62)
      IF (F(N50,N63).EQ.0) GOTO202
      DO 203 N64=1,MGR
      IF (F(1,N63).NE.GRAPH(N64,1)) GOTO 203
      CONTINUE
      DO 204 N65=1,TR
      IF(GRAPH(N64,2).NE.TRUNK(M8,N65)) GOTO 204
      TGR=TRUNK(M8,N65)
      IF(GRAPH(N64,3).EQ.1) POSONE=2
      CONTINUE
      IF(GRAPH(N64,3).EQ.2) POSONE=1
      GOTO 196
  204 CONTINUE
  203 CONTINUE
  202 CONTINUE
  196 MLEFT=50
      IF (CASE.EQ.6) TGR=0
      MRIGHT=0
      TPOS=0
      DO 205 N66=1,TR
      N67=TRUNBR(N66)
      IF (F(N50,N67).EQ.0) GOTO 205
      CONTINUE
      DO 206 N68=1,TR
      IF (TRUNK(M8,N68).EQ.TGR) TPOS=N68
      CONTINUE
      IF (TRUNK(M8,N68).NE.F(1,N67)) GOTO 206
      CONTINUE
      IF (N68.LT.MLEFT) MLEFT=N68
      CONTINUE
      IF (N68.GT.MRIGHT) MRIGHT=N68
  206 CONTINUE
  205 CONTINUE
      IF (CASE.EQ.6) GOTO 207
      CONTINUE
      IF (TPOS.EQ.MLEFT) GOTO 208
  207 MGR=MGR+1
      GRAPH(MGR,1)=F(N50,1)
      GRAPH(MGR,3)=1
      IF (MRIGHT.EQ.MLEFT.AND.MLEFT.EQ.TPOS)
     1 GRAPH(MGR,3)=POSONE
      GRAPH(MGR,2)=TRUNK(M8,MLEFT)
      IF (CASE.NE.6) GOTO 183
  208 MGR=MGR+1
      GRAPH(MGR,1)=F(N50,1)
      GRAPH(MGR,3)=2
      IF (MRIGHT.EQ.MLEFT.AND.MLEFT.EQ.TPOS)
     1 GRAPH(MGR,3)=POSONE
      GRAPH(MGR,2)=TRUNK(M8,MRIGHT)
  183 CONTINUE
      RETURN
      END


      SUBROUTINE LIMBS
C THE LIMBS ARE CONNECTED SO AS TO FORM LINEAR
C  PATHS WITH MAIN OR TRUNK BRANCHES
       IMPLICIT INTEGER (A-Z)
      DIMENSION F(30,30),LIMB(20),NOTLIM(20),MAINBR(20)
      DIMENSION TRUNBR(20)
      DIMENSION MEND(30),GRAPH(60,3),TRUNK(20,20),FPART(30)
      COMMON/LLO/ ROW,COL,F,TR,TRUNBR,M8,TRUNK,MBR,MAINBR,
     1MEND,MGR,GRAPH,L,LIMB,ME
      CONTINUE
      DO 177 N42=2,ROW
      ROWLIM=0
      DO 176 N40=1,L
      N41=LIMB(N40)
      IF (F(N42,N41).EQ.0) GOTO 176
      ROWLIM=ROWLIM+1
      IF (ROWLIM.EQ.1)GOTO 219
      MGR=MGR+1
      GRAPH(MGR,1)=F(1,N41)
      GRAPH(MGR,2)=GRAPH(MGR-1,1)
      GRAPH(MGR,3)=7
      GOTO 176
 219  NOMBR=0
      MSTORS=0
      DO 178 N43=1,MBR
      N44=MAINBR(N43)
      IF (F(N42,N44).EQ.0) GOTO 178
      NOMBR=1
      DO 179 N45=1,ME
      IF (F(1,N44).NE.MEND(N45)) GOTO179
      CONTINUE
      IF (N45.LT.MSTORS) GOTO 179
      MSTORS=N45
 179  CONTINUE
 178  CONTINUE
      IF (NOMBR.EQ.0) GOTO 180
      MGR=MGR+1
      GRAPH(MGR,1)=F(1,N41)
      GRAPH(MGR,2)=MEND(MSTORS)
      GRAPH(MGR,3)=3
      GOTO 176
 180  MSTORT=50
      NOTTBR=0
      DO 181 N46=1,TR
      N47=TRUNBR(N46)
      IF (F(N42,N47).EQ.0) GOTO 181
      CONTINUE
      DO 182 N48=1,TR
        IF (F(1,N47).NE.TRUNK(M8,N48)) GOTO 182
      NOTTBR=1
      IF (N48.GT.MSTORT) GOTO 182
      MSTORT=N48
 182  CONTINUE
 181  CONTINUE
      MGR=MGR+1
      GRAPH(MGR,1)=F(1,N41)
      GRAPH(MGR,2)=0
      GRAPH(MGR,3)=6
      IF (NOTTBR.EQ.0) GOTO 176
      GRAPH(MGR,2)=TRUNK(M8,MSTORT)
      GRAPH(MGR,3)=11
 176  CONTINUE
 177  CONTINUE
      RETURN
      END


      SUBROUTINE CHAINS(K)
C CHAINS ARE CHECKED FOR DUPLICATIONS
        IMPLICIT INTEGER (A-Z)
      DIMENSION KSTORE (20),KCHAIN(20,10,20),KC(10)
      COMMON /CH/ KCHAIN,KSTORE,KC
      IF (KC(K).EQ.0) GOTO 1
      KR=KC(K)
      DO 2 K1=1,KR
      KSAME=0
      DO 3 K2=1,K
      K3=KCHAIN(K,K1,K2)
      DO 3 K4=1,K
      IF (K3.EQ.KSTORE(K4)) KSAME=KSAME+1
 3    CONTINUE
      IF(KSAME.EQ.K) GOTO 5
 2    CONTINUE
 1    KC(K)=KC(K)+1
      KR=KC(K)
      DO  4  K3=1,K
 4    KCHAIN (K,KR,K3)=KSTORE(K3)
 5    RETURN
       END


      SUBROUTINE JTRONE (N,T,*)
C DELETE THOSE JTRCH'S WHICH HAVE BEEN COMBINED,
C  ALSO DELETE THEIR VARIATIONS
       IMPLICIT INTEGER (A-Z)
      DIMENSION JTRCH(30,30),LENGTH(20),JLAST(10),JFIRST(10)
      DIMENSION JTEMP (30)
      DIMENSION JADD(30,9)
      COMMON/PT/JTRCH,LENGTH,JTWO,JLAST,JFIRST,JADD,
     1JTEMP ,JT,JF,JL
      IF (N.EQ.0) GOTO 1
      DO 2 J1=1,N
      IF (IABS(LENGTH(J1)).NE.IABS(LENGTH(T))) GOTO 2
      J2=IABS(LENGTH(T))
      DO 3 J3=1,J2
      IF (JTRCH(J1,J3).NE.JTRCH(T,J3)) GOTO 1
 3    CONTINUE
 2    CONTINUE
 1    RETURN
 4    RETURN1
      END


      SUBROUTINE PERMUT (J,NEW,*)
C GENERATE VARIATIONS OF NEW JTRCH'S SUCH
C  THAT PERMUTABLE BRANCHES APPEAR ONCE IN
C   FIRST OR LAST POSITION OF A JTRCH
       IMPLICIT INTEGER (A-Z)
      DIMENSION JTRCH(30,30),LENGTH(20),JLAST(10),JFIRST(10)
      DIMENSION JTEMP (30)
      DIMENSION JADD(30,9)
      COMMON/PT/JTRCH,LENGTH,JTWO,JLAST,JFIRST,JADD,
     1JTEMP ,JT,JF,JL
      IF (J.EQ.1)  GOTO 12
      CONTINUE
      IF (J.EQ.2)  GOTO 11
      CONTINUE
      IF (J.EQ.3)  GOTO 3
      CONTINUE
      IF (J.EQ.4)  GOTO 10
      CONTINUE
      IF (J.EQ.5)  GOTO 5
      CONTINUE
      IF (J.EQ.6)  GOTO 6
 3    JTURN=JFIRST(1)
      JF=0
      JL=1
      JLAST(1)=JTURN
      GOTO 8
 5    JTURN=JLAST(2)
      JF=1
      JL=1
      JFIRST(1)=JTURN
      GOTO 8
 6    JTURN=JFIRST(2)
      JF=1
      JL=1
      JLAST(1)=JTURN
 8    J1=LENGTH(JTURN)
      IF (J1.LT.0) GOTO 10
      J2=J1/2
      DO 9 J3=1,J2
      JTP=JTRCH(JTURN,J3)
      JTRCH(JTURN,J3)=JTRCH(JTURN,J1-J3+1)
 9      JTRCH(JTURN,J1-J3+1)=JTP
10      CONTINUE
        IF (JT.NE.0) GOTO 11
        NEW=0
        GOTO 16
11      CONTINUE
        IF (JT.GT.1) GOTO 12
        NEW=1
        JADD(1,1)=JTEMP(1)
        GOTO 16
12      NEW=0
        DO 13 J4=1,JT
        IF (J   .EQ.1.AND.((J4+1)/2).EQ.(J4/2)) GOTO 13
        NEW=NEW+1
        JCOUNT=0
        DO 14 J5=1,JT
        JR=J4+J5-1
        IF (JR.GT.JT)JR=JR-JT
        JCOUNT=JCOUNT+1
        JADD(NEW,JCOUNT)=JTEMP(JR)
        IF (J.EQ.1.OR.JTWO.EQ.0) GOTO 14
        JNEW=NEW+JT
        JADD(JNEW,JCOUNT)=JTEMP(JT+1-JR)
14      CONTINUE
13      CONTINUE
        IF (J.EQ.1) GOTO 15
        CONTINUE
        IF (JTWO.EQ.0) GOTO 16
        NEW=JNEW
        GOTO 15
15      RETURN
16      RETURN1
        END


        SUBROUTINE NODUP
C DELETE THOSE OF JTRCH'S WHICH ARE DUPLICATED
C   OR WHICH VIOLATE A COMMON NODE REQUIREMENT
          IMPLICIT INTEGER (A-Z)
        DIMENSION JTRCH(30,30),LENGTH(20),JLAST(10),JFIRST(10)
        DIMENSION JTEMP (30)
        DIMENSION MBOTH(20),MNO(20),MLORF(20)
        DIMENSION JADD(30,9)
        COMMON /NOD/JDELET(40),JD,COMB(40),C,J
        COMMON/PT/JTRCH,LENGTH,JTWO,JLAST,JFIRST,JADD,
       1JTEMP ,JT,JF,JL
        C=0
        MB=0
        MO=0
        MLF=0
        DO 1 M1=1,J
        GO=0
        MLE=LENGTH(M1)
        DO 2 M2=1,JT
        IF(JTEMP(M2).EQ.JTRCH(M1,1)) GO=GO+1
        CONTINUE
        IF(JTEMP(M2).EQ.JTRCH(M1,MLE))GO=GO+1
 2      CONTINUE
        IF(GO-1) 3,4,5
 3      MO=MO+1
        MNO(MO)=M1
        GOTO 6
 4      MLF=MLF+1
        MLORF(MLF)=M1
        GO TO 6
 5      MB=MB+1
        MBOTH(MB)=M1
 6      CONTINUE
 1      CONTINUE
        IF (MB.EQ.0)GOTO 7
        DO 10 M5=1,MB
        M6=MBOTH(M5)
```

```
      IF(LENGTH(M6).LT.0) GOTO 10
      M7=LENGTH(M6)
      IF (C.NE.0) GOTO 8
      C=C+1
      COMB(C)=M6
      LENGTH(M6)=-M7
      GOTO 10
8     CONTINUE
      DO 9 M3=1,C
      M4=COMB(M3)
      MTEST=JTRCH(M4,1)
      DO 13 M8=1,M7
      IF(JTRCH(M6,M8).NE.MTEST)GO TO 13
      JD=JD+1
      JDELET(JD)=M6
      LENGTH(M6)=-M7
      GOTO 10
13    CONTINUE
9     CONTINUE
      COMB(C+1)=M6
      LENGTH(M6)=-M7
      C=C+1
10    CONTINUE
7     CONTINUE
      IF(MLF.EQ.0)GOTO 14
      DO 17 M11=1,MLF
      M12=MLORF(M11)
      IF(LENGTH(M12).LT.0)GOTO 17
      M13=LENGTH(M12)
      IF (C.NE.0) GOTO 18
      COMB(C+1)=M12
      LENGTH(M12)=-M13
      C=1
      GOTO 17
18    CONTINUE
      DO 16 M9=1,C
      M10=COMB(M9)
      MTEST=JTRCH(M10,1)
      DO 20 M14=1,M13
      IF(JTRCH(M12,M14).NE.MTEST)GOTO 20
      JD=JD+1
      JDELET(JD)=M12
      LENGTH(M12)=-M13
      GO TO 17
20    CONTINUE
16    CONTINUE
      C=C+1
      COMB(C)=M12
      LENGTH(M12)=-M13
17    CONTINUE
14    CONTINUE
      IF(MO.EQ.0)GO TO 21
      CONTINUE
      DO 22 M15=1,MO
      C=C+1
22    COMB(C)=MNO(M15)
21    CONTINUE
      RETURN
      END
```

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	20
2.	Library Naval Postgraduate School Monterey, California 93940	2
3.	H.Q. Federal German Navy c/o Navy Section MAAG Germany APO 09080, New York	5
4.	Professor S. R. Parker Dept. of Electrical Engineering Naval Postgraduate School Monterey, California 93940	5
5.	Professor S. G. Chan Dept. of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
6.	Professor S. P. Chan Dept. of Electrical Engineering University of California, Santa Clara Santa Clara, California	1
7.	LCDR Hans J. Lohse, FGN 285 Bremerhaven Walter Deliusstr. 35a West Germany	3

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Naval Postgraduate School Monterey, California 93940	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP

3. REPORT TITLE

A PROCEDURE FOR THE SYNTHESIS OF A FUNDAMENTAL LOOP OR CUT SET MATRIX

4. DESCRIPTIVE NOTES *(Type of report and, inclusive dates)* Thesis for degree of "Electrical Engineer"

5. AUTHOR(S) *(First name, middle initial, last name)*

Hans J. LOHSE
Lieutenant Commander, Federal German Navy

6. REPORT DATE December 1968	7a. TOTAL NO. OF PAGES 128	7b. NO. OF REFS 19

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. DISTRIBUTION STATEMENT

Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Naval Postgraduate School Monterey, California 93940

13. ABSTRACT

Existing realization procedures for a fundamental loop or cut set matrix are reviewed, compared, and classified broadly on the basis of their underlying approach. A new combinatorial synthesis technique is presented utilizing the concepts of trunk branches, main branches, limbs, and unique connections which are introduced. This procedure is direct, easy to apply and learn, general, and yields an expression for the number of physically different or alternate realizations which are possible. A general computer program for realization of the graphs is presented and illustrated with some examples.

UNCLASSIFIED
Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Procedure						
Synthesis						
Fundamental Loop Matrix						

DD FORM 1 NOV 65 1473 (BACK)

UNCLASSIFIED
Security Classification

A-31409

Gaylord
SHELF BINDER
Syracuse, N. Y.
Stockton, Calif.

thesL7914
A procedure for the synthesis of a funda
3 2768 001 03357 4
DUDLEY KNOX LIBRARY